UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Implementación de un sistema HIL en tiempo real basado en un SoC

Máster Universitario en Ingeniería de Telecomunicación

Autor: BENEDICTO RIDAURA, Jorge
Tutor: SÁNCHEZ GONZÁLEZ, Alberto
Ponente: DE CASTRO MARTÍN, Ángel

Julio, 2020

Implementación de un sistema HIL en tiempo real basado en un SoC

AUTOR: Jorge Benedicto Ridaura
TUTOR: Alberto Sánchez González
PONENTE: Ángel de Castro Martín

Dpto. de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2020

Resumen

En los últimos años se están produciendo grandes avances en la electrónica de potencia gracias al uso creciente de las energías renovables. Dichos avances también son producidos por la evolución del control digital, que está adquiriendo una mayor importancia con respecto al control analógico. Esto es debido a que el control digital permite diseñar algoritmos más complejos, con capacidad de reprogramación y con mucha más fiabilidad que el control analógico.

Como cualquier otro producto ingenieril, pero especialmente debido a la gran cantidad de energía que puede manejarse, un regulador de un convertidor de potencia debe sufrir unas exhaustivas simulaciones antes de ser probado con la planta real. Sin embargo, la naturaleza analógica de la planta, unido al control digital, da lugar a un sistema mixto complejo de simular. Por tanto, tradicionalmente existieron varias opciones de simulación, donde destaca la simulación mixta utilizando simuladores que soportan esquemáticos y código digital, pero cuyo rendimiento es muy bajo.

Una alternativa que está adquiriendo mucha popularidad en la última década son las técnicas HIL (*Hardware In The Loop*), que se basan en incluir elementos hardware que emulen el comportamiento de la planta a simular. De esa forma, el simulador HIL emulará el convertidor de potencia y el usuario podrá probar su control digital con dicha planta emulada. Estas técnicas tienen cada vez más importancia puesto que se llegan a conseguir simulaciones en tiempo real con pasos de simulación de centenas, e incluso decenas, de nanosegundos. Esta gran mejoría se ha conseguido con el uso de FPGAs (*Field-Programmable Gate Arrays)*, gracias a que así se pueden realizar múltiples tareas en paralelo, consiguiendo reducir el paso de integración.

En este trabajo se pretende desarrollar un sistema HIL completo donde se diseñe e implemente no solo el modelo digital del convertidor conmutado, sino también dotarlo de comunicación con una aplicación de escritorio que permita configurarlo. El objetivo es realizar todas las capas de un sistema HIL: creación del modelo matemático del convertidor, un convertidor de potencia Flyback, dotar al sistema de comunicación mediante un protocolo TCP/IP y creación de una aplicación de escritorio con la que configurar dicho modelo. Como elemento adicional se ha creado un capturador de datos digital (a modo de osciloscopio) con el fin de visualizar los datos obtenidos en las simulaciones. Este sistema está basado en una arquitectura SoC (System on Chip) que integra una FPGA y un microprocesador ARM, donde la primera se utilizará para la creación del simulador en tiempo real, mientras que la segunda se utilizará para dotar de comunicaciones al simulador, sirviendo de intermediario entre el simulador y la aplicación de escritorio.

Palabras clave

Control digital, convertidor de potencia, Hardware in the Loop, Field-Programmable Gate Arrays, Flyback.

Abstract

Power electronics has experimented great advances in recent years thanks to the increasing use of renewable energies. These advances are also produced by the evolution of digital control. This control is acquiring more important than analogic control. Digital control allows us to design more complex algorithms, with reprogramming capabilities and much greater reliability than analogic control.

Like any other engineering product, but especially because of the large amount of energy that can be handled, a power converter regulator must undergo exhaustive simulations before being tested with the real plant. However, the analogic nature of the plant joined to digital control leads to a mixed system, very difficult to simulate. Traditionally, there were several simulations options but the most important to stand out is a mixed simulation which uses a simulator that supports schematics and digital code, but whose performance is very low.

Despite the wide variety of options available, one alternative that has become very popular in the last decade is the HIL (Hardware in The Loop) techniques. These techniques were based on including hardware elements that emulate the behavior of the plant to be simulated. The HIL simulator will then emulate the power converter and the user will be able to test its digital control with the emulated plant. This alternative is becoming more and more popular because they enable you to get real time simulations with simulation steps of hundreds, even tens, of nanoseconds. This great improvement has been achieved with the use of FPGAs (Field-Programmable Gate Arrays), whose board allows you to do multiple tasks performed in parallel, succeeding in reducing the integration step.

The aim of this work is to develop a complete HIL system where we design and implement not only the digital model of the switched converter, but also provide it with a desktop application to be capable of configuring this model. Last but not least, one of the main purposes of this work is to develop all the layers of a HIL system: the creation of the power converter, a Flyback power converter, to provide the communication system through a TCP/IP protocol and to create a desktop application with which to configure this model. As an additional element, a digital data capture device has been created (like an oscilloscope) to display the data obtained in the simulations. This system is based on an SoC (System on Chip) architecture that integrates an FPGA and an ARM microprocessor. The architecture will be used for creating the simulator in real time, while the microprocessor will be used to provide communication to the simulator, serving as an intermediary between the simulator and the desktop application.

Keywords

Digital control, power converter, Hardware in the Loop, Field-Programmable Gate Arrays, Flyback.

Agradecimientos

En primer lugar quería agradecer a mi tutor, Alberto Sánchez, por brindarme la oportunidad realizar este trabajo al igual que el TFG. También agradecerle el apoyo, las correcciones y las constantes ayudas que me ha aportado durante este último año, incluso de forma telemática. También quería agradecer a Ángel de Castro por hacer que me interesase por los sistemas de control desde aquella primera clase de tercero.

Por supuesto agradecer a mi familia todo el apoyo recibido estos seis años. Tanto en la carrera como el máster siempre me habéis apoyado y ayudado cuando más lo necesitaba. En especial a mi madre que es la que me ha estado aguantando todos los días desde que empecé el primer día, ¡muchas gracias! Por supuesto, también a mi padre por estar a mi lado y siempre aconsejarme correctamente. Y a mi hermana por estar en el lugar correcto siempre que la he necesitado.

También agradecer a todos mis compañeros y amigos que siempre me han apoyado cada día, especial mención a Nacho con el que pasaba más tiempo que con mis padres.

Por último, este trabajo va dedicado a mi abuelo. Te queremos mucho.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción

1.1 Motivación

La generación de energías renovables está teniendo un gran avance gracias a la investigación e implementación de plantas fotovoltaicas y eólicas. Los avances en la electrónica de potencia, así como en el control digital (cada vez más avanzado) están produciendo que este campo tenga gran importancia en la última década. Dicho control digital está imponiéndose respecto al analógico debido a que, con el paso de los años, va adquiriendo una serie de ventajas, como encontrar la posibilidad de implementar algoritmos más complejos, la capacidad de reprogramación, la disminución de la sensibilidad a cambios (los componentes digitales no envejecen), etc. Sin embargo, también tiene inconvenientes como puede ser la necesidad de utilizar ADCs, limitación del ancho de banda y, sobre todo, el aumento del precio. Este último se está reduciendo notablemente con el abaratamiento de los componentes digitales [6].

El uso de los convertidores de potencia se ha generalizado con el fin de conseguir transformar la energía eléctrica de entrada a la salida con diferentes características. Por ejemplo, se pueden utilizar para manejar la tensión o corriente producidas por esas plantas fotovoltaicas.

Al igual que cualquier otro producto de ingeniería, un regulador digital debe ser exhaustivamente simulado antes de ser producido. Por una parte, puede ser peligroso, pudiendo producir incluso daños personales, ya que se mueve mucha energía. Por otra parte, porque la planta puede ser muy cara y no se quiere poner en peligro, o porque a veces, se diseña un controlador antes de que la planta exista físicamente.

Para realizar pruebas sobre esos convertidores de potencia hay tres modos posibles, la simulación analógica, la digital o la mixta. En el primer tipo se podría modelar la respuesta analógica del regulador y simular íntegramente con un simulador analógico, pero perdiendo gran realismo en la simulación. Otra opción es realizar una simulación mixta con un simulador que permita simular esquemáticos y código digital. Estos simuladores son muy lentos y están limitados en el número de lenguajes que soportan. Algunos permiten simular código C, o código VHDL, pero la libertad del tipo de regulador a probar está muy limitada. Una última opción sería realizar un modelo digital de la planta para poder simular todo el sistema de forma digital. De esta forma, se puede utilizar un dispositivo que emule el convertidor, creando un modelo digital de el mismo, que es la opción utilizada en este trabajo. Si la simulación digital se ejecuta en un ordenador, ésta sigue siendo lenta.

Para intentar mejorar dichas simulaciones la tecnología HIL (Hardware-in-the-loop) está avanzando a pasos agigantados ya que con esta tecnología se puede emular en hardware un convertidor de potencia para que el usuario pueda probar su regulador digital en tiempo real. Una característica importante de los simuladores HIL es que permiten depurar el controlador en su implementación final, ya que dicho regulador ya implementado puede ser conectado físicamente al hardware de emulación, independientemente del lenguaje o tecnologías usados para su implementación.

Los sistemas HIL actuales utilizan FPGAs (*field-programmable gate arrays*) con el fin de realizar numerosos cálculos en paralelo, reduciendo así el paso de integración. En herramientas comerciales, la opción de utilizar FPGAs está adquiriendo una gran importancia debido a que así se pueden realizar tareas en paralelo e intentar conseguir que la simulación sea en tiempo real.

Para diseñar el circuito hay dos tipos de aritmética, la coma fija y la coma flotante. La coma fija adquiere mayores velocidades y utiliza una menor cantidad de recursos, sin embargo, es complicada para diseñar el convertidor ya que el usuario debe establecer la posición de la coma para cada variable, además de que el diseño no puede adaptarse automáticamente. Esto provoca que, si el usuario cambia el valor de una variable, tendrá que modificar la ubicación de la coma manualmente. Sin embargo, la coma flotante es más lenta pero para diseñar el convertidor es mucho más sencillo debido a que adapta la posición de la coma de forma nativa y automáticamente. Por tanto, si las prestaciones lo permiten, la coma flotante es preferible en muchas ocasiones.

1.2 Objetivos

El principal objetivo de este proyecto es la realización de un sistema HIL completo de un convertidor de potencia sencillo, que va a ser el convertidor Flyback. La plataforma utilizada será una Zynq, que es un SoC (system on chip) de la compañía Xilinx que integra una FPGA y un microprocesador empotrado (ARM). En nuestro caso, se utiliza lógica programable para el modelo y el procesador para la comunicación con la aplicación de ordenador. Para ello, no solo es necesario diseñar e implementar el modelo digital, sino también permitir la comunicación con una aplicación de ordenador que configure el sistema. Esto permitirá realizar multitud de pruebas para obtener el funcionamiento óptimo del circuito. El protocolo de comunicación será mediante ethernet, por medio de un protocolo TCP/IP.

Mientras que el modelo digital será realizado en la lógica programable de la FPGA, la comunicación con el ordenador hará uso del procesador empotrado. Para comprobar los datos obtenidos, el sistema ofrece los resultados mediante un osciloscopio digital en el que el usuario podrá configurar un *trigger*, además de configurar cada cuanto tiempo se desea capturar unas muestras y observar de forma precisa el comportamiento de las señales. Por otra parte, los datos podrán ser extraídos mediante DACs (digital to analog converters), para que el sistema HIL se comporte igual que el sistema emulado. Por tanto, el proyecto se dividirá en dos bloques:

- El primer bloque consiste en el diseño e implementación del modelo del convertidor Flyback, utilizando la aritmética de coma flotante. Para las señales se utilizará el modelo IEEE-754 con un tamaño de señal de 32 bits. Se integrará el convertidor en la FPGA incluyendo los protocolos de comunicación convenientes para mandar la configuración del modelo que no tiene requisitos temporales estrictos, pero también de grandes volúmenes de datos como serían los datos del osciloscopio digital. Para ello, dado que el procesador es ARM, se evaluarán los protocolos de la especificación AMBA.
- El segundo bloque será la aplicación de ordenador; en él se crea un formulario mediante el lenguaje Java en el que el usuario introducirá los valores de configuración del Flyback. Una vez configurado, se mostrará en dicho formulario una visualización similar a un osciloscopio digital del funcionamiento de dicho

convertidor. Esa visualización seguirá unos parámetros de captura de los datos configurable por el propio usuario también en el formulario. La información presente en el osciloscopio debe ser trasladada mediante técnicas eficientes de trasvase de información, como puede ser DMA (*Direct Memory Access*).

Una vez se hayan juntado los dos bloques, se podrán realizar multitud de pruebas del convertidor de potencia y observar su comportamiento.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- En el segundo capítulo se muestran los tipos de convertidores de potencia y los avances realizados hasta el día de hoy en ese ámbito, además de nombrar los distintos tipos de control y algunas herramientas comerciales.
- En el tercer capítulo se detalla la forma de implementar el diseño del convertidor de potencia, el tipo de coma utilizado, y la arquitectura que va a tener el convertidor. Se muestran los protocolos de captura y envío de datos desde el ordenador al procesador de la FPGA, así como ecuaciones que modelan el convertidor Flyback.
- En el cuarto capítulo se muestra la integración del periférico creado (convertidor + comunicación) en la FPGA, además de explicar el protocolo ethernet utilizado para comunicar la FPGA con el ordenador.
- En el quinto capítulo se muestra el software creado en el microprocesador Zynq para leer y enviar valores. También el software desarrollado para la aplicación de ordenador donde se configurará el convertidor, la forma de captura de los datos y por último, el osciloscopio digital creado donde se mostrarán los valores obtenidos en el convertidor.
- En el sexto capítulo se muestran los resultados obtenidos mediante una serie de simulaciones creadas. Se comparará el convertidor creado en coma flotante contra un convertidor creado con señales de tipo *real*, además de ver el efecto del interruptor de entrada del circuito en la simulación.
- Por último, se mostrarán las conclusiones obtenidas al realizar este trabajo y los posibles futuros trabajos que puede tener.

2 Estado del arte

En los últimos años se está observando cómo el mundo digital está imponiéndose sobre el mundo analógico. Como no puede ser de otra manera, en el entorno de los sistemas de control y convertidores de potencia está ocurriendo los mismo. Esto es debido a que las ventajas del control digital respecto al analógico son cada vez mayores, ya que proporciona una mayor flexibilidad y ofrece mayores posibilidades [1-2].

Actualmente hay sistemas que requieren tener un control realizando pequeños y constantes cambios para conseguir tener un funcionamiento óptimo. Un ejemplo de sistema de control sería un coche autónomo, el cual tiene multitud de sensores y sistemas de control como la velocidad de crucero. En ambos casos, el usuario establece el valor que desea mantener (como por ejemplo la velocidad) y el sistema aumentando o disminuyendo la potencia tiene que ser posible de mantener ese valor. Por ejemplo, en un coche autónomo habría multitud de sensores como podría ser aumentar o disminuir la velocidad, la frecuencia de los limpiaparabrisas, etc.

2.1 Convertidores de potencia

Los convertidores de potencia son dispositivos capaces de controlar voltajes/corrientes de entrada y modificarlos por otros de salida, previamente delimitadas por el usuario. Los convertidores se pueden dividir en cuatro categorías respecto al tipo de corriente que utilizan [3]:

- Los convertidores CA-CC o rectificadores son los encargados de transformar la corriente alterna en continua. Un ejemplo de ellos serían los convertidores usados para PFC (Power Factor Correction), que rectifican la corriente de entrada pero introduciendo a la vez muy poco contenido armónico en la red.
- Los convertidores CC-CC, se pueden usar por ejemplo en suministros de energía y aplicaciones de accionamientos motrices con el fin de elevar o reducir la tensión de acuerdo a la referencia introducida por el usuario.
- Los convertidores CC-CA o inversores. Este tipo de convertidor se puede utilizar, por ejemplo, en motores de corriente alterna y fuentes de alimentación no interrumpibles. También se utiliza en los paneles fotovoltaicos, que producen una tensión CC, la cual hay que introducirla en la red y por tanto, para transportarla, hay que convertirla en CA.
- Por último, los convertidores CA-CA se utilizan para modificar el valor eficaz de la tensión de entrada, por ejemplo, para conseguir que el arranque de un motor sea lento y después ir aumentando el voltaje.

Sin embargo, no solo los convertidores de potencia pueden llevar a cabo la transformación de la energía eléctrica. Los reguladores lineales son otro dispositivo que también pueden realizar dicha transformación. Estos reguladores basan su funcionamiento en la disipación de energía por medio de calor, por ello se utilizan resistencias variables. A diferencia de los reguladores, los convertidores hacen uso de interruptores (a veces son sustituidos por MOSFETs) que permiten o niegan el paso del voltaje proporcionado desde la fuente de alimentación. Debido a que los reguladores

lineales utilizan la disipación de calor para rebajar la tensión, este tipo de dispositivos no son recomendables cuando se utilizan valores de voltajes de entrada muy altos, por lo que para entradas de 40 W o más no se recomienda su uso [4]. Además, otra ventaja de los convertidores es que no se hace uso de resistencias para dejar que la tensión baje, sino que son capaces de reducir la tensión, tal y como hace el convertidor de tipo CC-CC, *Buck*.

Los convertidores de potencia tienen diferentes maneras de conmutación; la primera de ellas es la conmutación natural (también llamada convertidores de frecuencia en línea), en la que las tensiones de la línea de alimentación presentes en un lado del convertidor facilitan la conexión o desconexión de los dispositivos semiconductores. Otra manera de conmutación es la conmutación forzada (también llamados convertidores de conmutación), donde los interruptores del convertidor se conectan y desconectan con frecuencias altas en comparación con la frecuencia presente en el circuito, con el fin de conseguir el voltaje deseado. Por último, se encuentran los convertidores resonantes y cuasirresonantes, donde los interruptores se conectan o desconectan con tensión cero [3].

Una vez se han explicado los tipos de conmutación, el usuario debe decidir el tipo de control que quiere realizar, si analógico o digital. Ambos tienen sus ventajas y desventajas, aunque poco a poco el control digital va adquiriendo una mayor importancia debido a que se están modernizando las técnicas utilizadas sin aumentar en gran medida su precio. Debido a ese precio, que es muy inferior al control analógico, cada vez más usuarios se inclinan por el control digital. La importancia de depurar controladores para convertidores de potencia es fundamental para evitar, por ejemplo, daños materiales debido a altos voltajes o tensiones inesperadas. Por tanto, para evitarlos es ideal hacer un control digital con el que probar el convertidor, sin embargo, hay una serie de problemas para simular el circuito analógico. El primero y fundamental es que estos simuladores son extremadamente complejos de realizar, ya que simular en tiempo real un circuito analógico mediante control digital tiene una gran complicación.

Los controladores analógicos pueden implementarse con elementos circuitales reales con los que va a ser posible generar señales visibles a tiempo real. Aunque el control digital está adquiriendo mayor importancia, los controladores analógicos tienen algunas ventajas sobre los digitales como puede ser su alta resolución, un alto ancho de banda y una de las más importantes, que son levemente fáciles de diseñar y verificar [5-6].

Por otra parte, los controladores digitales suelen hacer uso de microcontroladores o microprocesadores. Para lograr leer los valores obtenidos en la planta hacen uso de conversores ADC y DAC, cuyos componentes pueden adquirir retardos en la simulación. La resolución no es tan alta con el controlador analógico pero es bastante preciso, por el contrario, diseñarlo resulta mucho más complicado. Sin embargo, dichos controladores se pueden hacer configurables con lo que el usuario puede realizar multitud de pruebas sin tener que modificar el hardware [6].

Debido a que los convertidores son elementos analógicos, se hace necesario realizar simulaciones mixtas (analógicas y digitales a la vez). Para realizar una simulación mixta se dan tres opciones; la primera consiste en crear un modelo analógico del control, sin embargo, esta opción es inviable en la mayoría de los casos. La segunda opción es utilizar un simulador lo suficientemente potente como para entender el lenguaje del

regulador, aunque todavía no se ha desarrollado tal simulador puesto que el regulador puede ser un programa de un ordenador, un microcontrolador, una Arduino, una FPGA, etc [6]. La tercera opción sería utilizar un dispositivo que emule el convertidor, creando un modelo digital de él, que es la opción elegida para realizar este trabajo. Un tipo de simulación mixta sería el utilizado en [7], donde se utiliza el simulador PSIM para la parte analógica, mientras que para parte digital se utiliza el simulador ModelSim. Sin embargo, esto no es fácil ya que se tienen que enlazar dos programas y sincronizar los datos [9].

Como se ha explicado anteriormente, existen varios tipos de convertidores, los cuales se dividen en dos grandes grupos; los convertidores aislados y los no aislados. Los no aislados disponen de una etapa de aislamiento eléctrico, los más utilizados serían el circuito elevador (*Boost*), el reductor (*Buck*) y el reductor/elevador (*Buck-Boost*) que sería una combinación de los anteriores. Estos convertidores se corresponden con los convertidores de tipo CC-CC. Los circuitos aislados utilizan dispositivos como transformadores para aislar una parte del circuito de la otra, un ejemplo de este circuito sería el Flyback. Dicho convertidor tiene las mismas condiciones que el circuito reductor/elevador pero con aislamiento, el cual permite que dependiendo de las condiciones del interruptor-MOSFET de entrada, el circuito elevará o reducirá la tensión entrante al mismo.

Otra forma de diseñar el circuito es modelar la planta en lenguaje HDL a partir de las ecuaciones en diferencias que modelan el circuito. En [8] se realiza una comparación entre VHDL (lenguaje que se utiliza en circuitos digitales) y VHDL-AMS (lenguaje que se utiliza en circuitos mixtos). En esa referencia se comprueba que el tiempo de ejecución de los circuitos digitales, es decir, del lenguaje VHDL, es mucho menor que el tiempo de ejecución de un circuito mixto, es decir, del lenguaje VHDL-AMS.

2.2 Hardware in the Loop (HIL)

Otra tecnología que está avanzando a pasos agigantados es la tecnología HIL (Hardware-in-the-loop) que se ha convertido en una técnica viable de prueba para la electrónica de potencia. HIL permite al diseñador probar, incluso en tiempo real, el controlador junto con la planta. Una característica muy importante de los simuladores HIL es que permiten depurar el controlador en su implementación final, es decir, usando la misma plataforma que realmente se utilizará en el convertidor final. Esto es posible porque el sistema HIL es un dispositivo externo con las mismas entradas y salidas que tendrá el convertidor real [9].

En [10] se exponen una serie de alternativas de simulación de microrredes, y manifiesta el correcto uso de HIL en electrónica de potencia. Los primeros sistemas se basaron en ordenadores, obteniendo pasos de simulación de 50 µs [11]. Sin embargo, actualmente los sistemas HIL utilizan FPGAs (*field-programmable gate arrays*) debido a que su naturaleza paralela permite hacer numerosos cálculos en procesamiento paralelo, por lo que el paso de simulación cae drásticamente hasta conseguir decenas de nanosegundos [12-13].

Otra opción muy común, que además es la utilizada en este trabajo es la de usar la FPGA junto con los procesadores para crear los sistemas HIL. Cuando se hace uso de esta opción, la FPGA se utiliza para realizar tareas que se pueden ejecutar en paralelo

como muestreo y acondicionamiento de señal entrada/salida, mientras que el sistema utiliza procesadores en tiempo real [14]. En la misma referencia se muestran como las herramientas comerciales Opal-RT y dSPACE utilizan ambos dispositivos en sus sistemas. El problema de dichos simuladores es su alto precio, por ejemplo, pueden llegar a costar más de 6.000€ debido a que implementar el modelo es una tarea muy compleja. La implementación de un código HDL (*hardware description language*) no es nada sencillo. El tipo de aritmética que se utilice dentro del modelo decidirá una serie de importantes factores como son el paso de la simulación, la facilidad del diseño y los recursos utilizados.

Otra posibilidad es diseñar el modelo con herramientas de alto nivel, como los modelos de MATLAB, y después, traducirlos a código HDL utilizando herramientas automáticas. También hay herramientas como Typhoon HIL que le permiten al diseñador crear modelos gráficos y usarlos para la simulación con la ayuda de la FPGA. El problema fundamental de dichas herramientas de nuevo es el precio, pueden llegar a costar más de US $ 10.000 . Esto es debido a que estos sistemas pueden llegar a manejar modelos muy complejo, definidos por el usuario, con pasos de integración de hasta cientos de nanosegundos. Por tanto, estas herramientas solo llegarán a ser rentables para modelos muy complejos. En [15] se utiliza la plataforma OPAL-RT para verificar el controlador de un cargador rápido de vehículos eléctricos. los sistemas comerciales HIL facilitan la implementación del modelo, sin embargo, los resultados de rendimiento no son óptimos, tanto en recursos como en pasos de simulación [9].

2.3 Tipos de aritmética

Un gran problema del diseño del modelo es la elección de la aritmética que se va a utilizar en el modelo, en la librería de VHDL hay diferentes opciones para tratar los datos. Las dos formas más importantes son coma fija y coma flotante.

Sin embargo, una alternativa sería utilizar el tipo de datos *real*, que implementa coma flotante de doble precisión (64 bits). Ya que es coma flotante, no hay que preocuparse por la posición de la coma ya que se coloca de forma nativa. Sin embargo este tipo de aritmética no es implementable en la FPGA y, por tanto, no puede ser sintetizado en hardware real. Sin embargo, se puede realizar una simulación software mediante herramientas como Questa Sim.

La aritmética de tipo coma fija es implementable en la FPGA, sin embargo, el usuario debe situar el valor de la coma en cada variable. Además el usuario debe delimitar el tamaño de todas las variables dependiendo del valor que tendrá cada una, lo que resulta tedioso si debes realizar algún cambio ya que tendrías que modificar todos los tamaños. El formato de esta aritmética es QX.Y. Donde X se corresponde con los bits de parte entera e Y con los bits de parte decimal, además de un bit adicional al principio delimitando el signo de la variable.

La aritmética en coma flotante es mucho más sencilla de utilizar ya que permite de forma nativa mover la posición de la coma en cada variable sin que el diseñador se preocupe. Para esta aritmética se emplean señales de 32 bits en las que el primer bit equivale al signo de esta (0-positivo, 1-negativo), los 8 bits siguientes corresponden al exponente y los 23 restantes a la mantisa. Este tipo de aritmética también es

implementable en la FPGA y, de hecho, es la que suelen usar las principales herramientas HIL comerciales debido a que la ubicación de la coma se cambia dinámicamente dependiendo del valor que posea dicha variable.

La gran diferencia entre los dos últimos tipos de aritmética a simple vista es si el usuario debe ubicar la posición de la coma en las variables o se realiza de forma nativa. Sin embargo, hay otras diferencias importantes. Utilizando la coma fija, el uso de recursos desciende de manera sustancial, aumentando a su vez la velocidad. En [16] se muestra que la velocidad de la coma flotante puede llegar hasta 10 veces más lenta que la coma fija. En [17], se propone el uso de la librería *float_pkg* en el lenguaje VHDL-2008 para HIL. En ese caso se utiliza un microprocesador Microblaze para el controlador obteniendo unas simulaciones muy largas. El problema de esta librería es que de momento no es compatible con todas las herramientas de síntesis (en nuestro caso, con Vivado 2018.3) [16].

3 Modelo HIL y Osciloscopio Integrado

3.1 Convertidor Flyback

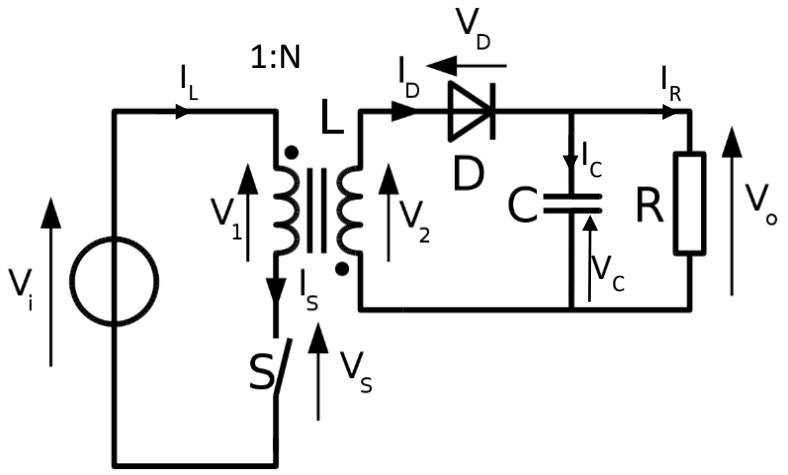

Figura 3-1. Convertidor Flyback

En este capítulo se va a describir el convertidor que se va a modelar mediante sus ecuaciones en diferencias, la manera de realizarlo y el tipo de aritmética utilizado. Además, se explicarán los protocolos de comunicación necesarios para lograr enviar y recibir datos de la aplicación de ordenador.

Un convertidor Flyback está compuesto por numerosos elementos, entre los que destacan un condensador, un interruptor y un transformador. El interruptor en particular es el elemento central del sistema puesto que con su acción se controlará el convertidor del sistema (este normalmente será un MOSFET como es el caso de este proyecto). Este circuito, como se explicará más adelante, puede tanto elevar como reducir la tensión de salida del circuito, por lo que hace las mismas funciones que un *Boost* (circuito elevador) y un *Buck* (circuito reductor), con la ventaja respecto a dichos circuitos de que la entrada y salida están aisladas, ya que, como se ve en la Figura 3-1, hay un transformador. La elección del funcionamiento de circuito va a depender de los tiempos de encendido y apagado del interruptor. Para modelar el circuito se van a analizar las variables de estado de éste. Hay dos elementos con dinámica en el circuito: el transformador y el condensador. Viendo cómo evolucionan esos elementos, se modela el circuito.

En primer lugar, se obtiene la ecuación del transformador; éste se modela como una bobina con la inductancia equivalente del transformador, y una relación de vueltas entre primario y secundario 1:N, la variable de estado que atraviesa dicho componente es la corriente que circula por ella (i_L). Por tanto, la tensión que cae en la bobina sería la siguiente:

$$v_L = L * \frac{\partial i_L(t)}{\partial t} \qquad (3.1)$$

La ecuación 3.1 puede ser transformada en ecuación de diferencias para que su cálculo sea más fácil. Para ello se utilizan métodos ODE (*Ordinary Differential Equations*), y en particular en este trabajo se utilizará el método Euler explícito. Este es

un método sencillo de orden 1 pero que ofrece una precisión suficientemente alta cuando el paso de simulación es bajo. Utilizando dichos métodos es posible despejar la variable de estado i_L:

$$i_L(k) = i_L(k-1) + v_L(k-1)\frac{\Delta t}{L} \qquad (3.2)$$

Siendo Δt el paso de integración para calcular las variables de estado, k el instante de tiempo actual, k-1 el instante anterior, v_L la tensión que cae en la bobina y L la inductancia de la bobina. Se realiza el mismo proceso con el condensador donde la capacidad del condensador será C y la variable de estado a analizar es el voltaje del condensador v_C:

$$i_C = C * \frac{\partial v_c(t)}{\partial t} \qquad (3.3)$$

Al igual que en el proceso anterior, el valor del voltaje se despeja ya que es la variable de estado que se va a analizar:

$$v_C(k) = v_C(k-1) + i_C(k-1)\frac{\Delta t}{C} \qquad (3.4)$$

Otra parte importante del circuito es el interruptor, que provoca que se eleve o disminuya la tensión de salida v_{out} presente en el circuito. Cuando el interruptor está cerrado, va a hacer que la entrada y salida estén totalmente asiladas. Sin embargo, cuando el interruptor se abre, la energía almacenada en el transformador se traspasará a la parte secundaria del circuito. Por tanto, dependiendo de la situación del interruptor-MOSFET se dan dos situaciones completamente opuestas:

MOSFET = ON	**MOSFET = OFF**	
	$\boldsymbol{i_L > 0}$	$\boldsymbol{i_L = 0}$
$\boldsymbol{i_C = -i_R}$	$i_C = \frac{i_L}{n} - i_R$	$i_C = -i_R$
$\boldsymbol{v_L = v_G}$	$v_L = -\frac{v_C}{n}$	$v_L = 0$

Tabla 3-1. Estados del MOSFET y los valores de las variables de estado

Una vez se conoce cómo quedan las variables de estado del circuito se pueden desarrollar las ecuaciones por completo:

- Cuando el interruptor está conectado:

$$v_C(k) = v_C(k-1) + [-i_R(k)]\frac{\Delta t}{C} \qquad (3.5)$$

$$i_L(k) = i_L(k-1) + v_G(k)\frac{\Delta t}{L} \qquad (3.6)$$

- Cuando el interruptor está desconectado y la intensidad de la bobina es >0:

$$v_C(k) = v_C(k-1) + \left[\frac{i_L}{n} - i_R(k-1)\right] * \frac{\Delta t}{C} \qquad (3.7)$$

$$i_L(k) = i_L(k-1) - \frac{v_C}{n}(k-1)\frac{\Delta t}{L} \qquad (3.8)$$

- Cuando el interruptor está desconectado y la intensidad de la bobina es = 0:

$$v_C(k) = v_C(k-1) + [-i_R(k-1)] * \frac{\Delta t}{C} \qquad (3.9)$$

$$i_L(k) = i_L(k-1) = 0 \qquad (3.10)$$

Las ecuaciones mostradas son las que modelan el convertidor. Para poder entender el comportamiento en régimen permanente, se analizará la corriente del transformador en el primario (i_L). Si dicha corriente se encuentra en régimen permanente, el comportamiento de esta es similar a la Figura 3-2. Ahí se observa que la corriente se incrementa cuando el interruptor está cerrado y disminuye cuando el interruptor se abre. Si se encuentra en régimen permanente, el valor de la corriente será igual al inicio y al final del ciclo de conmutación del interruptor en un circuito ideal.

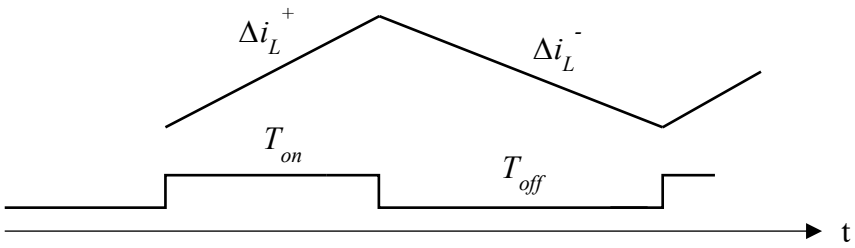

Figura 3-2. Comportamiento de la corriente del transformador en el primario

La modulación PWM (Pulse Width Modulation) se usa habitualmente en electrónica de potencia ya que se controla el convertidor con una frecuencia constante, pero cambiando la relación entre el apagado y encendido del interruptor. De esa forma, el convertidor se comporta como se desee. Esta relación se llama ciclo de trabajo y se define de la siguiente manera:

$$d = \frac{T_{on}}{T_{on} + T_{off}} \qquad (3.11)$$

Por tanto, según la ecuación anterior, el ciclo de trabajo es la proporción de la señal en encendido respecto al tiempo total. Una vez se dispone del valor del ciclo de trabajo, es posible desarrollar el comportamiento que tiene este sistema en régimen permanente. Después de la ecuación 3.10, se deduce que $v_{out} = v_c$. Por tanto, en condiciones de estabilidad se cumple necesariamente la siguiente ecuación:

$$\Delta i_L^+ + \Delta i_L^- = 0 \qquad (3.12)$$

Como se puede observar en la ecuación 3.12, el incremento de i_L debe anularse con el decremento, siempre y cuando sea régimen permanente. Los incrementos dependen del valor del interruptor como se vio en la Tabla 3-1, y de los tiempos T_{on} y T_{off} del interruptor. De la ecuación 3.11 se puede despejar el tiempo que el interruptor está desconectado $T_{off} = T_{on} * (1\text{-}d)$ y el tiempo que el interruptor está conectado $T_{on} = T_{off} / (1\text{-}d)$. Gracias a despejar dicha variable se pueden sustituir valores en la ecuación anterior para el caso general, es decir, cuando $i_L > 0$:

$$v_G * \frac{\Delta T}{L} * d * T + [-\frac{v_{out}}{n} * \frac{\Delta T}{L} * (1-d) * T] = 0 \qquad (3.13)$$

Al tener despejados los valores incrementales de la intensidad se puede desarrollar la fórmula con el fin de poder despejar el valor de la tensión de salida o tensión del condensador:

$$v_G * d - \frac{v_{out}}{n} * (1-d) = 0 \qquad (3.14)$$

Una vez se han retirado los valores comunes para ambos es posible despejar el valor de v_{out}:

$$v_G = \frac{v_{out}}{n} * \frac{1-d}{d} \rightarrow v_{out} = v_G * \frac{n*d}{1-d} \qquad (3.15)$$

Visto el resultado de la ecuación 3.15 se observa que hay infinitas posibilidades para el valor del voltaje de salida, sin embargo, hay dos casos donde el comportamiento del modelo es extremo, además de un caso intermedio que se mostrará a continuación:

$$\begin{gathered} d = 0 \rightarrow v_{out} = 0\,; \\ d = 0.5 \rightarrow v_{out} = v_G * n\,; \\ d = 1 \rightarrow v_{out} = \infty \end{gathered} \qquad (3.16)$$

Tal y como se ha dicho anteriormente, este convertidor puede realizar la misma función que un circuito *Buck o Boost* dependiendo del valor de ciclo de trabajo que utilice. Para mostrar dicho funcionamiento se utilizarán intervalos:

Valor del ciclo de trabajo en forma de intervalo	Tipo de convertidor
$\boldsymbol{d = (0, 0, 5)}$	Convertidor buck/reductor
$\boldsymbol{d = (0.5, 1)}$	Convertidor boost/elevador

Tabla 3-2. Intervalos del convertidor dependiendo del duty cicle

3.2 Modelo real en VHDL

Para comprobar el funcionamiento práctico del circuito se realiza una primera versión en VHDL con variables de tipo *real*. La realización de este circuito es muy sencilla en este tipo de variables ya que no se tiene que escoger el tamaño de la señal, sino que el software del sistema lo realiza automáticamente gracias a que *real* implementa el tipo coma flotante de doble precisión (64 bits), por lo que no hace falta preocuparse por el lugar que ocupa la coma, además de que la precisión es muy alta.

Este modelo no será implementable en la FPGA; solo se realizará la simulación en el ordenador ya que las variables de tipo *real* no son sintetizables. Puesto que no se puede probar dicho modelo en hardware real; se realizará una simulación software con el programa Questa Sim con el objetivo de comprobar las ecuaciones propuestas para el correcto funcionamiento del modelo. Con el fin de realizar el menor número de operaciones aritméticas dentro del código, al modelo se le introducirán directamente los valores *dtL* y *dtC* que equivalen a los términos $^{\Delta T}/_{L}$ y $^{\Delta T}/_{C.}$ Los resultados de esta simulación se mostrarán en la sección 6.2.

3.3 Modelo float VHDL

Una vez se tiene el convertidor Flyback implementado y sabiendo que las fórmulas están correctamente propuestas y despejadas, es el momento de pasar a otro formato de variable. El formato que se va a utilizar es el modelo IEEE-754 o modelo en coma flotante [18]. En VHDL la diferencia con el modelo anterior es la posibilidad de sintetizar (que el anterior no tenía); además de que este modelo utiliza una precisión simple. Se va a diseñar utilizando señales con un tamaño de 32 bits. Aunque sea menor precisión que para el diseño anterior, para la mayoría de las aplicaciones es suficiente. Se utiliza esta cantidad de bits debido a que con 64 bits se requiere el uso de una gran cantidad de recursos a veces inasumible.

Con el formato que se va a utilizar es posible realizar operaciones utilizando notación científica en binario. Este estándar define tres formatos binarios de longitudes distintas, de 32, 64 y 128 bits. En este trabajo se va a utilizar el formato de 32 bits como se ha dicho anteriormente ya que es una precisión suficientemente alta para este tipo de aplicaciones. El formato del número en IEEE-754 sería el siguiente:

Signo (1 bit)	Exponente (8 bits)	Mantisa (23 bits)

Tabla 3-3. Formato número IEEE-754 de 32 bits

El estándar IEEE-754 también tiene valores excepcionales para valores no decimales que son NaN (Not a Number), ±0 y ±∞. A continuación, se muestran todos los valores excepcionales del modelo en coma flotante:

Signo	Exponente	Mantisa	Valor
0	1111111	00000000000000000000	+∞
1	1111111	00000000000000000000	-∞
0/1	1111111	Distinta de todo ceros (ej: 10101010011001100011)	±NaN
0	0000000	00000000000000000000	+0
1	0000000	00000000000000000000	-0

Tabla 3-4. Casos excepcionales del estándar IEEE-754

Todos estos casos se tienen en cuenta en este proyecto por si en algún momento el modelo se desestabiliza, sin embargo, no se debería dar el caso siempre y cuando los valores del convertidor sean los correctos. El estándar VHDL-2008 ofrece coma flotante de forma nativa. Sin embargo, algunos sintetizadores todavía no lo soportan. En este trabajo se utilizará el programa Vivado, el cual no soporta actualmente la biblioteca *float_pkg*, por tanto, se utilizan cores IPs (Intellectual Property). Estos cores tienen como entradas y salidas *standard logic vectors*, pero internamente operan con el estándar IEEE-754.

Estos bloques de operaciones están proporcionados por Xilinx en el que únicamente hay que configurar el tipo de operación a realizar. Para no utilizar más IPs de lo necesario aumentando así el uso de recursos, no se van a utilizar bloques que realicen las operaciones de restar o dividir. Para ello el valor que se va a introducir al modelo será el valor negativo en el caso de que tengamos que restar un número o el valor inverso en el caso de una división. Un ejemplo para este proceso sería el caso de la división $^{\Delta T}/_{C}$. Para realizar dicha división se utiliza un multiplicador al que le entraran los valores ΔT y $^{1}/_{C}$, obteniendo con ello $^{\Delta T}/_{C}$ y con lo que será posible ahorrarse un bloque.

Una vez se tiene el modelo creado, el siguiente paso es probarlo. Partiendo del modelo creado anteriormente, se hace un archivo de pruebas (testbench) con el fin de comprobar el correcto funcionamiento del modelo. Los resultados de dichas pruebas se mostrarán en la sección 6.2.

3.4 Arquitectura del periférico

Una vez implementado el convertidor, es el momento de realizar el protocolo de comunicación entre una aplicación de ordenador y la FPGA. En primer lugar, se procede a explicar brevemente la aplicación ya que en una sección posterior se explicará de forma más extensa. Se creará una interfaz gráfica que constará de un formulario en el que el usuario podrá controlar tanto los valores de configuración del modelo como del simulador. Primero el usuario configurará tanto el modelo como el osciloscopio digital, y mediante una conexión ethernet se enviarán los valores a la FPGA. Una vez se han recibido, el convertidor empieza a funcionar y devuelve unos valores que mediante conexión ethernet vuelven directamente a la aplicación, donde mediante un osciloscopio digital se podrá observar el comportamiento del convertidor. Conociendo la estructura de la conexión entre la aplicación y la FPGA, se creará un periférico en el que se transmitirán dichos datos; como hay dos comunicaciones, se dividirá en dos partes.

El bloque *Modelo Flyback* es el explicado en las sección 3.1. El bloque *Configurador* es el encargado de recibir la información desde el microprocesador ARM de la FPGA, que a su vez recibe dichos datos de la aplicación. Este bloque traspasa los valores v_G, *R*, *dtL*, *dtC*, etc. El convertidor calcula en tiempo real las variables de estado, y estos valores son mandados al exterior mediante DACs (*Digital Analog Converters*). Además, el periférico desarrollado ofrece la funcionalidad de osciloscopio, capturando en unas memorias los valores de las variables de estado (v_C, i_L). Una vez llenas dichas memorias de acuerdo con los valores configurados por el usuario, es posible trasladar estos datos desde el bloque *Oscilo* a la memoria DDR del sistema mediante la técnica DMA (*Direct Memory Address*).

Además de estos dos bloques se dispone de un bloque *Controlador*, el cual controla el funcionamiento global del sistema, define cuándo deben escribirse valores en las memorias, cuándo leerlas y cuándo enviarlas de nuevo a la aplicación. Todo ello se realiza gracias a una máquina de estados en la que se indican de acuerdo a unas condiciones cuando tiene que leer/escribir. El esquema del periférico completo sería el siguiente:

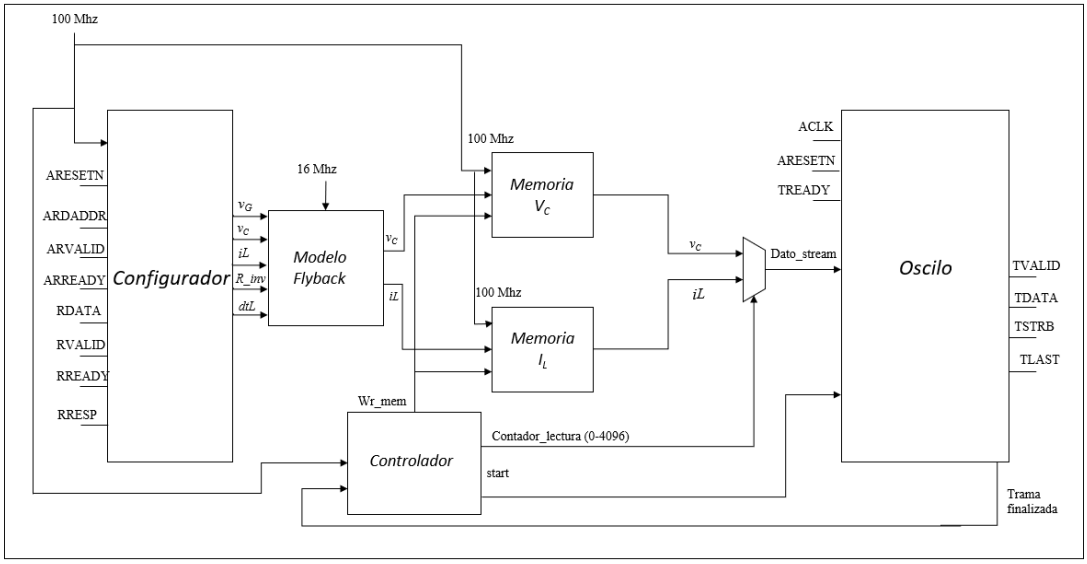

Figura 3-3. Esquema del periférico

3.4.1 Configurador

A la hora de configurar el modelo hay que incluir una serie de valores en las variables de configuración. Debido a que esto es un protocolo sencillo basado en la escritura de pocos registros no es necesario grandes prestaciones de velocidad. Debido a esto se utiliza el protocolo AXI- Lite [19]. Con este protocolo se pueden enviar un total de 64 variables de 32 bits cada una desde la aplicación directamente al modelo. Debido a que no es necesario incluir los 64 registros en el modelo, solo se usarán los estrictamente necesarios que dan un total de 13 registros. En esos 13 registros también se incluyen aquellas variables de configuración del osciloscopio digital, las cuales serán fundamentales a la hora de visualizar los datos obtenidos en el *Modelo Flyback*.

Número de registro	***Valor del registro***	***Número de registro***	***Valor del registro***
Registro 0	v_G	***Registro 1***	*R_inv*
Registro 2	$v_{C\_inicial}$	***Registro 3***	$i_{l\_inicial}$
Registro 4	*Load (solo el bit 0)*	***Registro 5***	i_{l_umbral}
Registro 6	v_{C_umbral}	***Registro 7***	*Modo trigger*
Registro 8	*dTL*	***Registro 9***	*dTC*
Registro 10	*n_inv*	***Registro 11***	*Trigger_tiempo_ muestra*
Registro 63		*Activación de la captura*	

Tabla 3-5. Valores de configuración

Como se puede observar en la Tabla 3-5, además de los valores de configuración del convertidor y del osciloscopio digital se incluye una señal en el último registro (63). Esta señal hará que el circuito se ponga a capturar el valor de las variables de estado para después mostrarlas en el osciloscopio digital. El funcionamiento del protocolo AXI-Lite sería el siguiente:

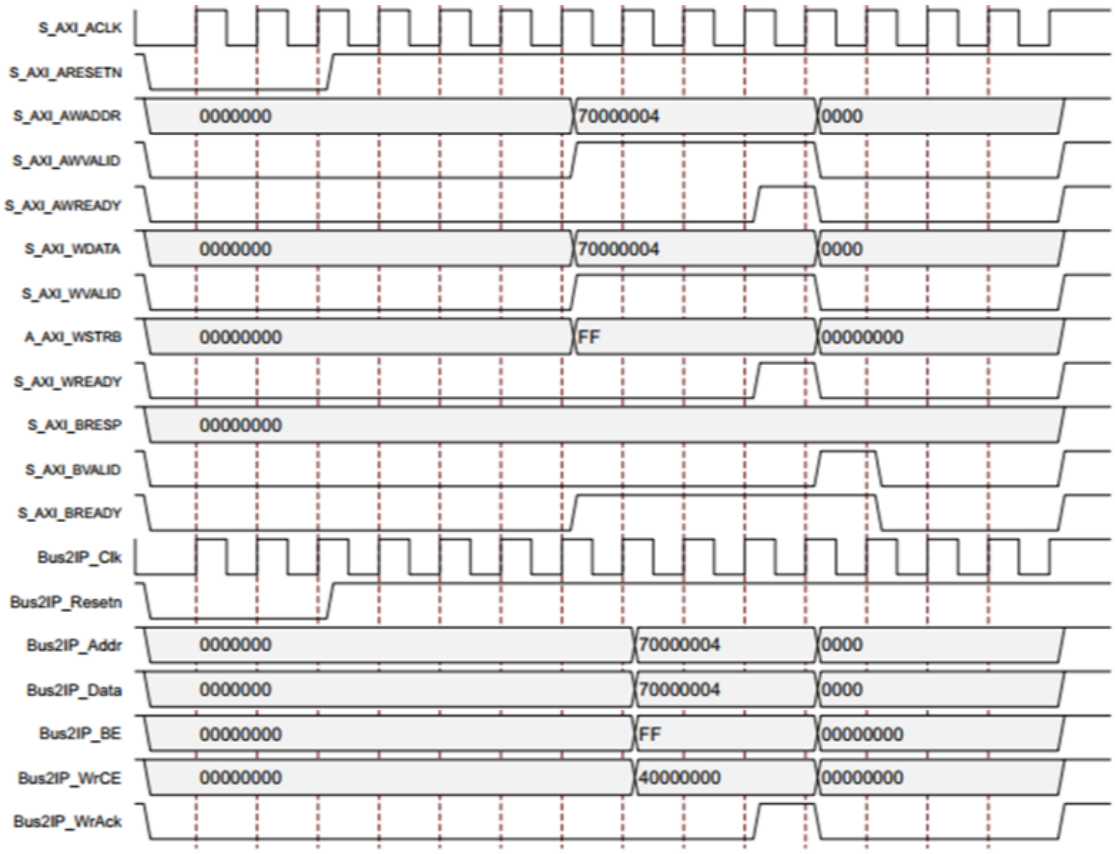

Figura 3-4. Funcionamiento del protocolo AXI-Lite [20]

Como se puede observar en la Figura 3-4, en este protocolo el maestro dice la dirección del registro que se va a escribir por S_AXI_AWADDR, e indica que es válida con S_AXI_AWVALID. En paralelo ofrece el dato a escribir con S_AXI_WDATA y dice que es válido cuando S_AXI_WVALID se pone a '1'. El esclavo tiene dos señales para confirmar la lectura de la dirección y dato, que son, S_AXI_AWREADY y S_AXI_WREADY. Cuando ambas se activan el dato será válido. Será en ese momento cuando se obtenga un registro de la Tabla 3-5 y se guarde en variables que después se utilizarán para el *Modelo Flyback* o en otros componentes.

3.4.2 Memorias

Debido a que en el osciloscopio digital únicamente se van a mostrar las variables i_L e v_C, habrá dos memorias. Estas son memorias de tipo Block RAM las cuales capturan datos en paralelo. En este caso se componen de un total de 2048 direcciones en donde se guardarán los datos y de las que después se podrán extraer sus valores. Como se va a explicar posteriormente, el periférico dispone de dos relojes. Las memorias van a funcionar con una velocidad de 100 MHz. Debido a esto surge un problema, ya que los valores de las memorias vienen desde el *Modelo Flyback* que funciona a una frecuencia

distinta, por tanto, se van a descartar valores. En la memoria se escriben datos múltiples veces en la misma dirección, hasta que el modelo genera un nuevo conjunto de valores válidos, dichos valores se escriben definitivamente en la memoria.

Puesto que el reloj del modelo va a 16 MHz, las memorias leerán 6 veces el mismo valor, quedándose con el último valor que coincide con el valor enviado desde el modelo, por tanto, los otros 5 valores se descartarán. Al tener dos memorias, se deben enviar ambas a la aplicación para poder leerlas, sin embargo, solo se ha creado un protocolo AXI-Stream. Para ello se crea un multiplexor controlado por el bloque *Controlador* que decide qué dato se envía antes o después. Para este proyecto, se ha creado de manera que primero se envíen los 2048 valores de i_L primero y cuando ya se han enviado al bloque *Oscilo*, se envían los 2048 valores restantes de v_c. En las posteriores secciones se explicará la forma de recepción de dichos valores en la aplicación. Por tanto, la trama en el AXI-Stream sería de la siguiente manera:

Dato a stream: $i_L[0]\ i_L[1]....i_L[1024]....\ i_L[2047]\ v_c[0]v_c[1]....v_c[1024]....\ v_c[2047]$

3.4.3 Oscilo

Una vez el *Modelo Flyback* está en funcionamiento es el momento de obtener los datos de salida con el fin de enviarlos a la memoria DDR para que se puedan transmitir a la aplicación del usuario, haciendo la funcionalidad de un osciloscopio (en nuestro caso digital) con el cual va a ser posible dibujarlo y observar el comportamiento de dicho circuito. Por ello no es posible solo enviar 64 registros de 32 bits como se ha realizado anteriormente, sino que se va a necesitar enviar una ráfaga de datos por cada captura entera de él para poder visualizar los datos de manera óptima y correcta. Por ello se utiliza un protocolo AXI-Stream [21]; el cual permite enviar flujos de datos de un tamaño determinado, en nuestro caso 32 bits, coincidiendo con el tamaño de un dato numérico del modelo (v_C, i_L).

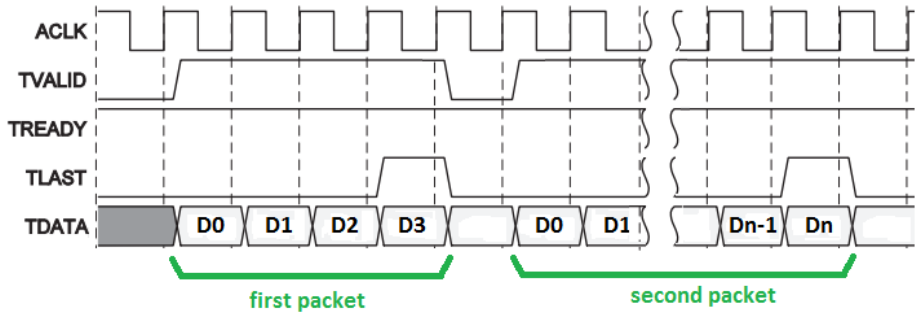

Figura 3-5. Funcionamiento del protocolo AXI-Stream [22]

Como se puede observar en la Figura 3-5, las variables importantes de control son TVALID, TLAST y TREADY; siendo esta última controlada por el esclavo. Se considerará que un dato es enviado si TREADY (del esclavo) y TVALID (del maestro) han estado a '1' en un ciclo de reloj. El TLAST es el encargado de indicar cuál va a ser el último paquete que se va a enviar. Cuando ya se ha enviado el último paquete, ambas variables se ponen a '0' hasta que se quiera volver a enviar otra cadena de valores. Este sencillo protocolo permite que el esclavo o maestro controlen la transmisión sin añadir

sobrecarga al protocolo. En este proyecto se va a enviar un total de 4096 valores por cada transmisión entre el bloque *Oscilo* y la memoria DDR del sistema.

3.4.4 Relojes internos

El modelo descrito en la sección 3.1 tiene una frecuencia máxima de funcionamiento de 16 MHz, generando dos salidas (v_C, i_L) en cada ciclo de reloj. Esas salidas se guardan en sendas memorias (block RAMs) y, cuando se han llenado las memorias, se transmiten en serie a la memoria DDR mediante el protocolo AXI-Stream. La lectura de las memorias, y su transmisión a la memoria DDR, se puede realizar más rápido, alcanzando los 100 MHz, reduciendo la latencia de dicha comunicación y dejando libre el osciloscopio para tomar otra muestra, esto permite obtener la disposición de dos relojes en el sistema, siendo el primero, de 100 MHz, proporcionado por el microprocesador y el segundo, creado mediante la herramienta *Clocking Wizard* de Xilinx con el que se consigue rebajar la frecuencia del reloj a los 16 MHz. Esta frecuencia se ha elegido con el fin de cumplir con los tiempos de *hold* presentes en los informes de Xilinx acerca de este proyecto.

Los procesos en los cuales existe la posibilidad de ir rápido son aquellos en los que no es necesaria ninguna operación aritmética, es decir, introducir los valores obtenidos por el *Modelo Flyback* en los bloques de memoria, obtener datos de la aplicación mediante el *Configurador* y, por último, enviar los valores a la aplicación mediante el *Oscilo*.

Sin embargo, el *Modelo Flyback* funcionará con el reloj lento, ya que para que tenga un correcto funcionamiento el reloj que debe tener el bloque es de 16 MHz. Al existir dos dominios de reloj, si no se produce una adaptación entre ellos, se puede producir la condición de metaestabilidad al no asegurarse los tiempos de *hold* y *setup*. Siendo el tiempo de *setup* la cantidad de tiempo que el dato debe ser estable a la entrada del Flip Flop antes del ciclo de reloj; y el tiempo de *hold* la cantidad de tiempo que el dato debe ser estable a la entrada del Flip Flop después del ciclo de reloj. Esto produciría que algunos valores capturados estén oscilando entre 1 y 0 durante un tiempo, provocando a su vez posibles fallos en los resultados del osciloscopio.

Por tanto, se puede reducir drásticamente la probabilidad de metaestabilidad registrando varias veces con el reloj de destino, como muestra la Figura 3-6.

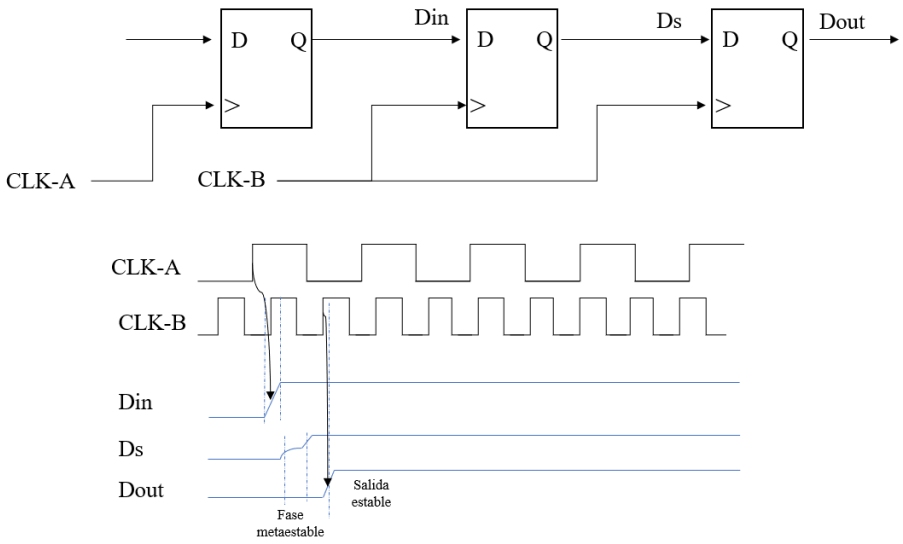

Figura 3-6. Ejemplo de doble registro

Como se puede observar en la Figura 3-6, se dispone de dos relojes CLK-A y CLK-B, ambos con una frecuencia distinta. Debido a esto el estado *Ds* se queda oscilando durante un tiempo, por lo que no se tiene la certeza del valor que otro dispositivo leerá y, lo que es peor, varios dispositivos podrían leer valores distintos debido a que esa salida es oscilante. Para ello la solución que se propone es la del doble registro, en el cual se guarda un valor no aleatorio y con ello se consigue disminuir el error al mínimo en los cálculos.

Este doble registro se ha utilizado a la entrada del valor de interruptor ya que la frecuencia a la que trabaja dicho proceso es distinta al reloj interno del modelo. Además, también se ha utilizado para los valores obtenidos en el *Modelo Flyback* (i_L y v_c). Utilizando el doble registro se consigue que el retraso de transmisión entre la salida del registro potencialmente oscilante y el siguiente sea mínima, incrementando así la posibilidad de que esa salida oscilante se estabilice y que el segundo registro capture el valor y se retransmita correctamente al resto de dispositivos que utilizan dicho valor.

Como se ha dicho anteriormente, debido a la complejidad del modelo no es posible realizar operaciones con mayor rapidez. Sin embargo, para que su funcionamiento sea el más rápido posible tanto las memorias como el *Oscilo* deben funcionan a una velocidad mayor. Esto permite que a una frecuencia de 16 MHz se generen datos del modelo en tiempo real, y que la transmisión a la memoria DDR se haga lo más rápido posible, es decir, a 100 MHz, minimizando la latencia del sistema.

3.4.5 Controlador

El bloque encargado de decir al sistema cuando ha de capturar es el controlador. Su finalidad es indicar el momento exacto en el que se cambia el envío de i_L a v_C, cuándo escribir los valores en la memoria y lo más importante, cuándo enviar una captura hacia la aplicación por medio del protocolo AXI-Stream. Todo el funcionamiento se realiza mediante contadores y una máquina de estados que se va a mostrar a continuación:

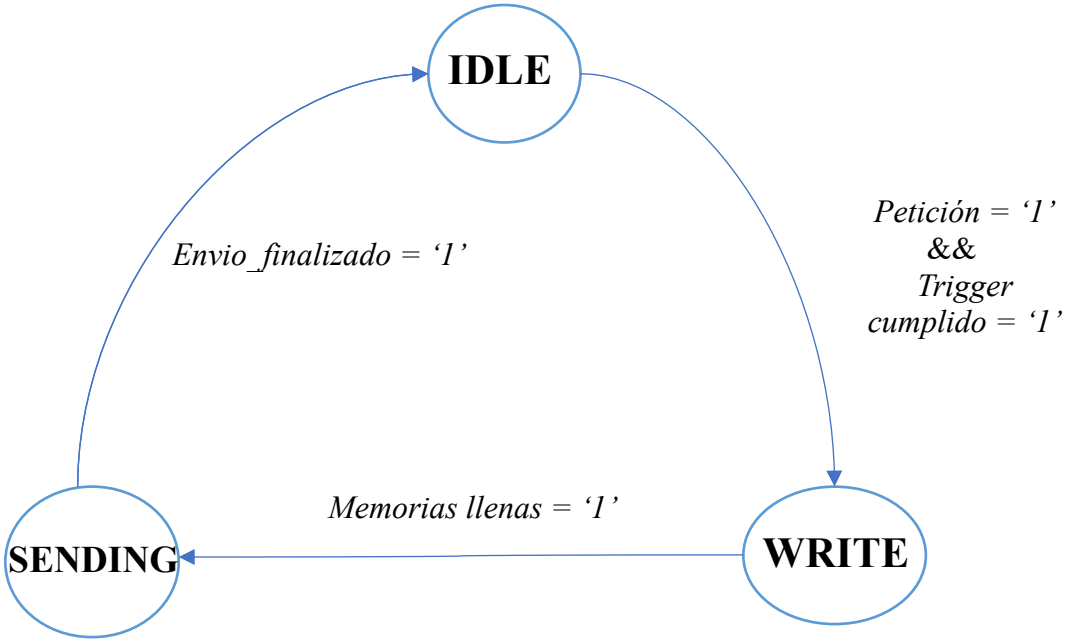

Figura 3-7. Máquina de estados del controlador

Al principio de la comunicación el estado inicial es *IDLE*. Una vez se configure el oscilo mediante los registros 5-6-7-11, hay un registro con el que se activa el osciloscopio, este es el registro 63. Este osciloscopio es totalmente autónomo y no tiene relación directa con el modelo. Es el usuario el encargado de "sincronizarlo". Al oscilo no le afecta la señal LOAD. Es decir, una vez se activa el START del *Oscilo* y cuando se cumpla la condición de *trigger*, se empezarán a rellenar las memorias y el sistema entrará en el estado *WRITE*.

En el estado *WRITE* se escriben los valores de las variables de estado en las dos memorias, como se ha indicado anteriormente, a una velocidad 100 MHz. Una vez las dos memorias estén llenas y el bloque *Oscilo* esté listo para recibir datos (que luego se enviarán a la aplicación) el sistema pasará al estado de envío, es decir, en *SENDING*.

Finalmente, en el estado *SENDING* se envían los valores desde las memorias a la aplicación de ordenador. Cuando se ha enviado por completo, el sistema vuelve otra vez al estado inicial (*IDLE*), del que no saldrá a menos que el usuario vuelva a pedir una captura de los datos.

4 Integración con Zynq

Una vez creado el modelo del convertidor y el osciloscopio digital, se añaden todos los protocolos de comunicación necesarios para integrar el modelo en la FPGA. Dicha FPGA es una placa de la compañía Digilent que integra una Zynq, concretamente el modelo de la FPGA es la Arty Z7-20. Dicha placa posee multitud de componentes de los cuales se usará el puerto ethernet para realizar una conexión ordenador-FPGA, el conector microUSB tanto como fuente de alimentación como para activar el funcionamiento del software de la FPGA, los LEDs como medida de comprobación, uno de los conectores de pines (*pmod ports*) para tener señales de entrada/salida y por último el procesador Zynq. La FPGA sería la siguiente:

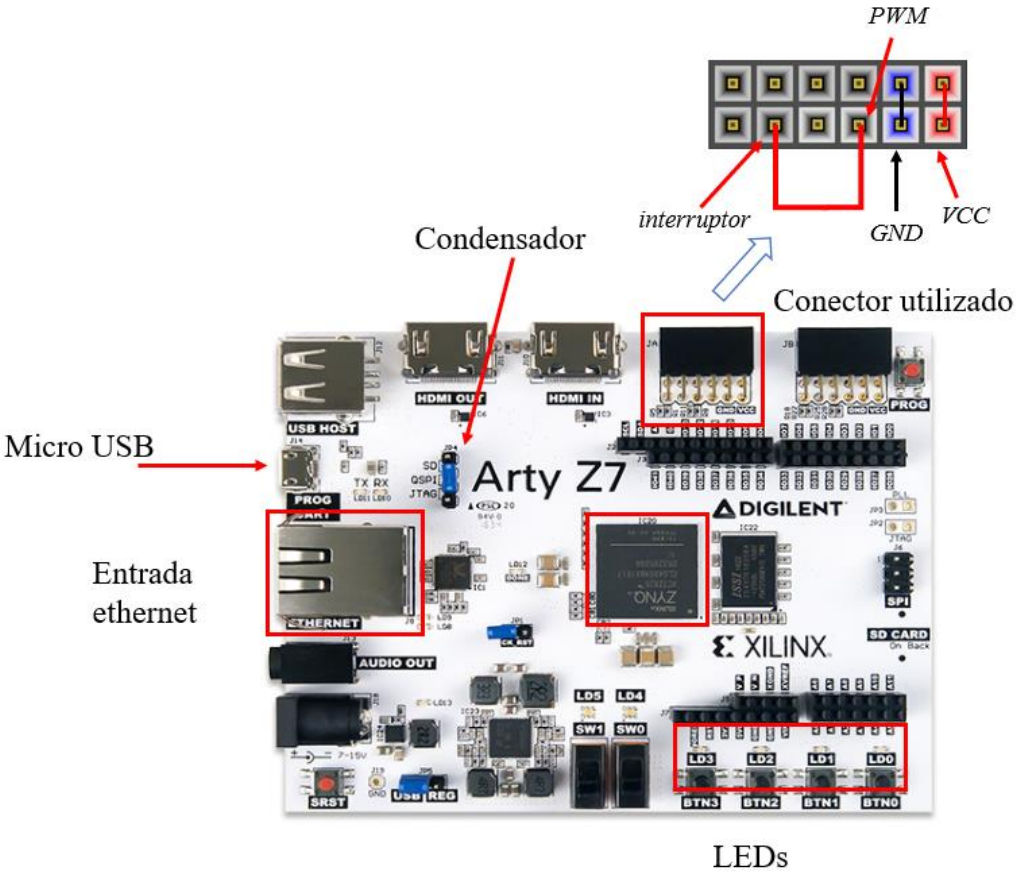

Figura 4-1. Arty Z7-20

En la Figura 4-1 se puede observar el lugar de cada componente en la placa. El condensador que indica la flecha estará situado conectando las señales QSPI y JTAG con el fin de poder programar la placa para así poder realizar numerosas pruebas. Las dos señales de entrada/salida del circuito global van a ser el interruptor y el PWM. Debido a esto con un conector es suficiente para realizar dichas conexiones. Más adelante se mostrará la configuración de los pines, sin embargo en la Figura 4-1 se muestra la colocación que tienen y como están conectadas entre sí.

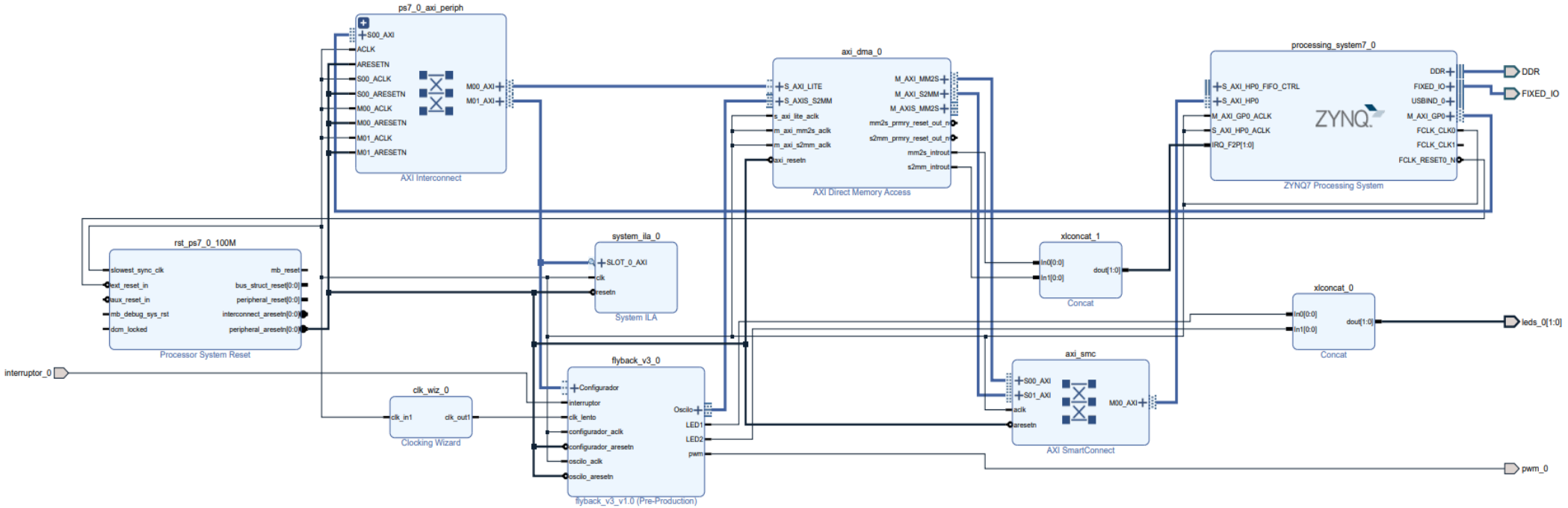

Figura 4-2. Esquema del proyecto completo

Visualizando la Figura 4-2 hay muchos componentes que se han nombrado en secciones anteriores como el *Clocking Wizard* o el periférico (flyback). El primero de ellos sirve para generar un reloj que se va a incluir en el *Modelo Flyback.* Se introduce el valor del reloj que genera por defecto la placa y se le indica el valor que se desea a la salida, en este caso 16 MHz, que como se ha indicado en la sección anterior, es la frecuencia máxima de funcionamiento del modelo del convertidor. Otro de los componentes es el periférico desarrollado en este trabajo. En él se puede observar las señales entrantes y salientes del bloque.

Como se ha explicado en la anterior sección, consta de dos partes, una en la que se recibe información y otra parte en la que se envía. El bloque *Configurador* utiliza una interfaz AXI-Lite y se conecta al microprocesador a través de un concentrador de tipo AXI Interconnect. Se observa cómo el valor del interruptor es una entrada del circuito, sirviendo dicha señal para controlar el convertidor de la misma forma que en un convertidor flyback real. Este valor será una entrada directamente a la FPGA; para ello hay que configurar un pin de entrada. También se van a utilizar los LEDs para comprobar que los valores de configuración se han recibido correctamente en la placa. Para conectarlos con entradas y salidas de la FPGA se ha de completar el fichero de *constraints*:

```
set_property IOSTANDARD LVCMOS33 [get_ports {leds_0[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {leds_0[0]}]
set_property PACKAGE_PIN R14 [get_ports {leds_0[1]}]
set_property PACKAGE_PIN P14 [get_ports {leds_0[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports interruptor_0]
set_property IOSTANDARD LVCMOS33 [get_ports pwm_0]
set_property PACKAGE_PIN Y19 [get_ports interruptor_0]
set_property PACKAGE_PIN Y17 [get_ports pwm_0]
```

Figura 4-3. Configuración de los pines

Como se puede observar en la Figura 4-3, el voltaje máximo tanto de los leds como del interruptor es de 3,3 V. Los pines han sido elegidos observando el esquemático de la FPGA y eligiendo un conector para todos ellos. También se ha utilizado el *System ILA*, cuyo bloque es utilizado con el fin de realizar pruebas de depuración sobre el DMA; consiguiendo con ello comprobar si los valores que se obtienen del convertidor se están enviando correctamente a las memorias. Con el *System Integrated Logic Analyzer* (ILA) se puede observar una serie de variables en tiempo de simulación de acuerdo con unas condiciones determinadas a elegir por el usuario. Más adelante en la sección de resultados se mostrará una imagen de las pruebas realizadas con el fin de comprobar el correcto funcionamiento de la integración.

4.1 Direct Memory Access (DMA)

El DMA proporciona un acceso entre la memoria DDR y el AXI-Stream con el cual el sistema podrá enviar datos a la aplicación por medio de una conexión ethernet [23]. Dicha memoria debe tener la capacidad de almacenar toda la cantidad de información obtenida a partir de los bloques de memorias i_L y v_C. Como en este proyecto se van a enviar un total de 4096 valores de 32 bits cada uno (2048 correspondientes a i_L y otros 2048 correspondientes a v_C). Esto da un total de 16 KB de información, por tanto,

hay que adaptar la capacidad de dicha memoria para que puedan guardar toda esa información en sus buffers.

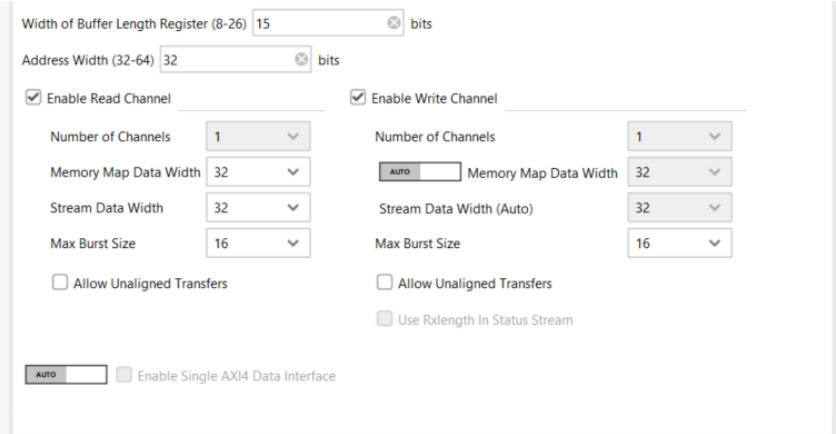

Figura 4-4. Configuración DMA

Para ello Xilinx permite configurar en la memoria el tamaño de la señal y el tamaño máximo de ráfaga, además de otros valores como se puede observar en la Figura 4-4. Debido a que se van a utilizar señales de 32 bits el tamaño de las señales será de 32, al igual que el tamaño máximo de la ráfaga será de 16. Como hay muchas formas de configuración, existe la posibilidad de aumentar dichas cantidades dependiendo de si se desea enviar más cantidad de bits. Esto producirá un aumento tanto en el uso de recursos como una disminución en la velocidad de envió. Como solo se va a utilizar un único protocolo *stream*, con un canal de comunicación es necesario. Con ello se consigue un uso óptimo de los recursos, consiguiendo así una menor velocidad, además de un menor uso de recursos.

4.2 Ethernet

Como modo de comunicación entre la FPGA y el ordenador se ha utilizado un cable ethernet. Se ha elegido dicha conexión debido a que se pueden alcanzar hasta 1 Gbps de velocidad dependiendo del cable utilizado. Con cables ethernet de categoría CAT5E o superior se consiguen velocidades de 1 Gbps, aunque a medida que se aumenta de categoría, la velocidad también es superior. Esto provoca que se puedan tanto enviar como recibir datos provenientes de la FPGA a velocidades mucho más altas que la que proporciona conectarlo por el puerto serie emulado a través del puerto microUSB que ofrece la placa. Sin embargo, para poder utilizar dicha comunicación se ha de crear una conexión estable entre el ordenador y la FPGA. Por consiguiente se han creado sockets con el fin de establecer una conexión TCP/IP [24].

Se ha utilizado el protocolo TCP/IP debido a que éste proporciona un transporte fiable de bits entre aplicaciones, en este caso, entre el ordenador y la FPGA. Esta parte requiere gran importancia puesto que no se puede perder ningún paquete en la conexión, ya que en ellos se incluyen tanto las variables de configuración del *Modelo Flyback* y *Oscilo*, como los valores que se envían de las memorias al ordenador con los valores correspondientes a i_L y v_C. Es por ello que perder uno de esos paquetes supondría un fallo provocando errores de configuración o de visualización. Una vez ya se han explicado la mayoría de los componentes que se utilizan, se procede a explicar el funcionamiento

global del sistema. Con el fin de que sea más sencillo primero se va a mostrar un esquema del mismo:

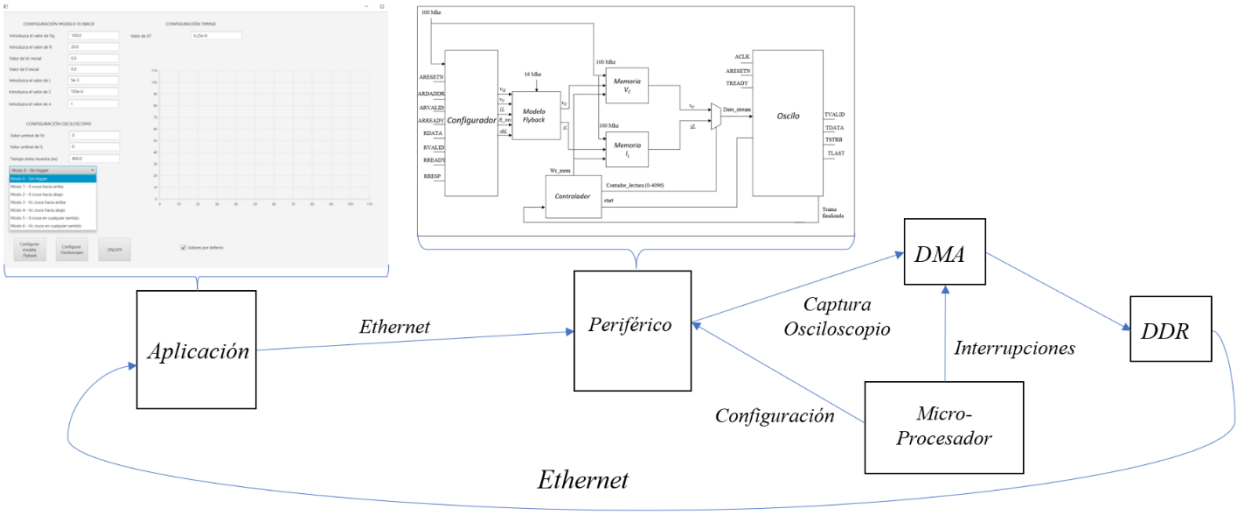

Figura 4-5. Esquema del funcionamiento global

El primer paso es la configuración del modelo, para ello el usuario tiene que introducir en la aplicación los valores de configuración tanto del *Modelo Flyback* como del componente *Oscilo* y enviarlo. Mediante la comunicación ordenador-FPGA vía cable ethernet se consiguen enviar los valores. Dicha configuración se recibe en la FPGA y, por medio del protocolo AXI-Lite, se consiguen obtener todos los valores, con los que se configurarán el *Modelo Flyback* y *Oscilo*. Una vez ya se disponen de todos los valores de configuración, el convertidor empieza a funcionar obteniendo valores a una velocidad de 16 MHz. Dicho funcionamiento es posible gracias al valor proporcionado por la variable de configuración LOAD, que sirve para cargar los valores iniciales de la simulación cuando es igual a '1' y arrancar la simulación cuando es igual a '0'.

El modelo realiza internamente todos los cálculos, y los envía a las memorias en paralelo a una velocidad de 100 MHz. Los valores que se envían a dichas memorias dependen de los valores de configuración del *Oscilo*. Hasta que alguno de los valores de las variables de estado (i_L o v_C) no supere las condiciones del *trigger*, los valores no se enviarán a las memorias. Finalmente una vez las memorias estén llenas, los datos se envían al bloque *Oscilo* cuando el controlador indique que se deben enviar los valores. Estos valores se enviarán directamente mediante DMA a las memorias DDR.

Debido a que en el sistema se utilizan interrupciones, hasta que no salte la interrupción correspondiente a que la transacción DMA está acabada (proporcionada por el microprocesador), no se podrán enviar los valores de vuelta a la aplicación para su visualización mediante un osciloscopio digital. Una vez los datos se envíen a la aplicación por medio de una trama TCP/IP, el usuario podrá visualizar correctamente el comportamiento que está teniendo el circuito y poder probar diferentes configuraciones con el fin de obtener el rendimiento óptimo. Si el usuario desea cambiar alguno de los valores simplemente tendrá que volver al principio del proceso enviando la configuración de nuevo a la FPGA.

5 Software

En las secciones anteriores se ha explicado tanto el funcionamiento como el hardware del sistema. Sin embargo, no se había nombrado el software que se ha desarrollado en el proyecto. En la parte de software hay dos partes delimitadas, la primera de ellas es la encargada de la comunicación entre el ordenador y la FPGA y la segunda de ellas es la aplicación de ordenador. En cuanto a la Zynq, la cual se corresponde con la primera parte, el programa Vivado ofrece un compilador ARM para el lenguaje C. En cuanto a la aplicación de ordenador, se utiliza Java, ya que es un lenguaje multiplataforma interpretado y potable, lo que facilita su ejecución en múltiples sistemas operativos y dispositivos.

5.1 Zynq

La programación software en la FPGA se realiza en el lenguaje C mediante el programa SDK de Vivado. Dicho programa cuenta con multitud de librerías seleccionadas donde se encuentran los periféricos estándar del chip Zynq, de los cuales se van a utilizar algunos como ethernet, DMA, timers, etc.

Los valores de configuración del convertidor se envían desde el ordenador hasta la FPGA por vía ethernet gracias al protocolo de conexión TCP/IP. Con el fin de facilitar la recepción de los valores de configuración, siempre se van a enviar ordenados de la misma manera. Un breve resumen sería: primero los valores del *Configurador*, después del *Oscilo* y, por último, el valor correspondiente a la variable LOAD, que sirve para reiniciar o reanudar la simulación.

La FPGA implementa un servidor TCP/IP. Para su correcto funcionamiento se deben configurar sus parámetros de IP, éstos son la dirección IP; la máscara de subred y la puerta de enlace, así como el puerto que estará a la escucha. Para realizar dicho servidor el archivo que se utilizó como base fue la aplicación *echo*, la cual viene como ejemplo de Xilinx. Una vez se disponía de la base se modificó con el fin de alcanzar el objetivo de este trabajo. Por tanto, una vez se ha modificado casi por completo el código de la aplicación *echo* se introducen los valores de configuración IP. En nuestro caso los valores utilizados son los siguientes:

- Dirección IP de la FPGA: **192.168.1.2**
- Mascara de Subred: **255.255.255.0**
- Gateway o Puerta de enlace: **192.168.1.1**
- Puerto: **7**

Una vez se configura la dirección IP de la placa y el puerto, se realizan las pruebas de la aplicación realizada. Para ello se activa el funcionamiento de la FPGA y después se habilita el socket mediante la aplicación de ordenador. Si se abre correctamente ya se pueden enviar los valores de configuración y recibir los datos sobre las variables de estado que se van a analizar. En caso de no abrirse correctamente, no se podrá enviar ningún valor de configuración. Esto puede ser debido a un fallo de configuración de la dirección IP o del puerto. Por tanto, el pseudocódigo de la aplicación final realizada en el chip Zynq sería el siguiente:

```
Configuración de Dirección IP de la FPGA {
        configuración IP_Addr;
        configuración Mascara de Subred;
        configuración Gateway (Puerta de enlace);
}
Configuración de DMA {
        inicializar el DMA;
        configuración tamaño mínimo de tramas
}
Configuración del periférico {
        configuración de la dirección del periférico
}
while (true) {
        Escucha del Puerto;
        if (flag_recepción ==1) {
                Switch (tipo configuración):
                        case 1: Recepción Valores Configurador
                        case 2: Recepción Valores Oscilo
                                Activación del sistema
                                Inicializar el buffer del AXI_DMA
                        case 3: Recepción Valor LOAD
        }
        if (flag_envio == 1) {
                Envío de los datos de la memoria a la aplicación;
        }
}
```

Como se puede observar en el pseudocódigo y como se ha explicado anteriormente, lo primero que hace el sistema es adquirir una dirección IP que asignarle a la FPGA. Dicha dirección IP será muy importante a la hora de abrir el socket para poder enviar y recibir los valores tanto del *Configurador* como de *Oscilo*. También se han de configurar las direcciones internas del periférico y la DMA. El DMA tiene una parte de configuración en la que se inicializa y se configura la capacidad de la memoria. Para que la conexión tenga suficiente memoria para almacenar dichos datos y enviarlos a la aplicación hay que configurar el tamaño del buffer TCP con el tamaño mínimo necesario (en este caso 20 KB). Una vez inicializado se comienza un bucle infinito del que nunca saldrá hasta que el usuario no desee realizar más pruebas y salga del programa.

En el caso de que sea una trama de recepción, es decir, una trama que proviene de la aplicación del ordenador se divide en tres partes, y éstas se delimitarán de acuerdo a un valor de configuración que se enviará al principio de cada trasmisión.

- Si el valor de configuración es 1 implica que los valores que se han enviado se corresponden con el bloque *Configurador*. Estos valores se enviarán directamente al periférico desarrollado en este trabajo por medio del protocolo AXI-Lite, como se ha explicado en secciones anteriores. En él se incluirán los valores iniciales del *Modelo Flyback* y las constantes de configuración del mismo. Dichos valores llegarán siempre siguiendo el mismo orden de llegada:

Orden de llegada	Variable recibida
Primero	V_G
Segundo	$^1/_R$
Tercero	$v_{C\_inicial}$
Cuarto	i_{L_incial}
Quinto	$^{dT}/_C$
Sexto	$^{dT}/_L$
Séptimo	$^1/_n$

Tabla 5-1. Orden de llegada de la configuración del convertidor desde la aplicación

- Si tiene valor 2 implica que los valores que se han enviado se corresponden con el bloque *Oscilo*. En él se envían los valores umbrales correspondientes al *trigger* que indicarán el punto exacto en el que se deben empezar a capturar datos al igual que en un osciloscopio; también se enviará el valor del tiempo entre muestra. El orden de llegada de dichos valores sería el siguiente:

Orden de llegada	Variable recibida
Primero	i_{L_umbral}
Segundo	v_{C_umbral}
Tercero	Modo del trigger
Cuarto	Tiempo entre muestra

Tabla 5-2. Orden de llegada de la configuración de la captura de los datos desde la aplicación

  Todos esos valores, al igual que los del bloque *Configurador* se enviarán al periférico por medio del protocolo AXI-Lite. Cuando se envíen estas variables se enviará además un bit en el campo 63 del protocolo que servirá como activación de la captura de los datos. Además, se realizará una petición de DMA con el fin de capturar los datos correspondientes a las variables de estado y se guardarán en un buffer a la espera de que esté completo para enviarlo a la aplicación de ordenador.
- Por último, si su valor es 3, se enviará el valor correspondiente a la señal LOAD. Como se ha explicado en secciones anteriores sirve para cargar los valores iniciales de la simulación cuando es igual a '1' y arranca la simulación cuando es igual a '0'.

Todos los valores provenientes de la aplicación vienen con el formato IEEE-754 para no tener que realizar ninguna conversión. Una vez se ha configurado todo el periférico, hay que esperar a que se cumplan las especificaciones del bloque *Oscilo* y que se envíen los valores desde la FPGA hacia el ordenador. Esto nos lleva directamente a la última fase de este software.

Una vez cumplido el *trigger*, el microprocesador configura una transferencia de tal forma que cuando se hayan calculado los valores, el DMA estará listo para enviar los datos a las memorias DDR para finalmente enviarlos a la aplicación. Para ello se ha de inicializar un buffer en el que se guardarán todos los valores con el tamaño de las variables, en este caso, un total de 4096 valores de tipo *uint32*. Cuando salte la interrupción que indica que se ha llenado la memoria DMA, será el momento en el que salte el *flag_envio*. Cuando esto ocurra, se mandarán todos los valores por medio de una conexión ethernet a la aplicación. Se envían en formato IEEE-754, por tanto, en la

aplicación habrá que traducirlos con el fin de conseguir el valor decimal. Dicha traducción se explicará en la siguiente sección.

5.2 Java

La aplicación de ordenador es una parte fundamental de este proyecto. Gracias a una conexión entre el ordenador y la FPGA se consiguen realizar distintas configuraciones de un mismo circuito con el fin de conseguir un rendimiento óptimo. Para ello, los usuarios que lo utilicen deben rellenar un formulario en el que se introduzcan los valores de configuración. Una vez se introduzca y active el circuito se podrá ver el comportamiento de dicho circuito por medio de un osciloscopio digital. En este proyecto se ha decidido realizar un osciloscopio digital con el fin de poder visualizar los valores sin tener que utilizar otros componentes. Puesto que la aplicación de ordenador consta de dos partes, se explicará primero la configuración y después la visualización de los valores.

Para ambas partes el protocolo de comunicación es el mismo, es decir, un protocolo TCP/IP. Para ello se ha de crear un socket en el que la dirección IP sobre la que se abre dicho socket es la dirección IP de la FPGA que se muestra al inicializar el funcionamiento de ésta. Esto permite obtener una comunicación con la que sea posible tanto enviar como recibir valores.

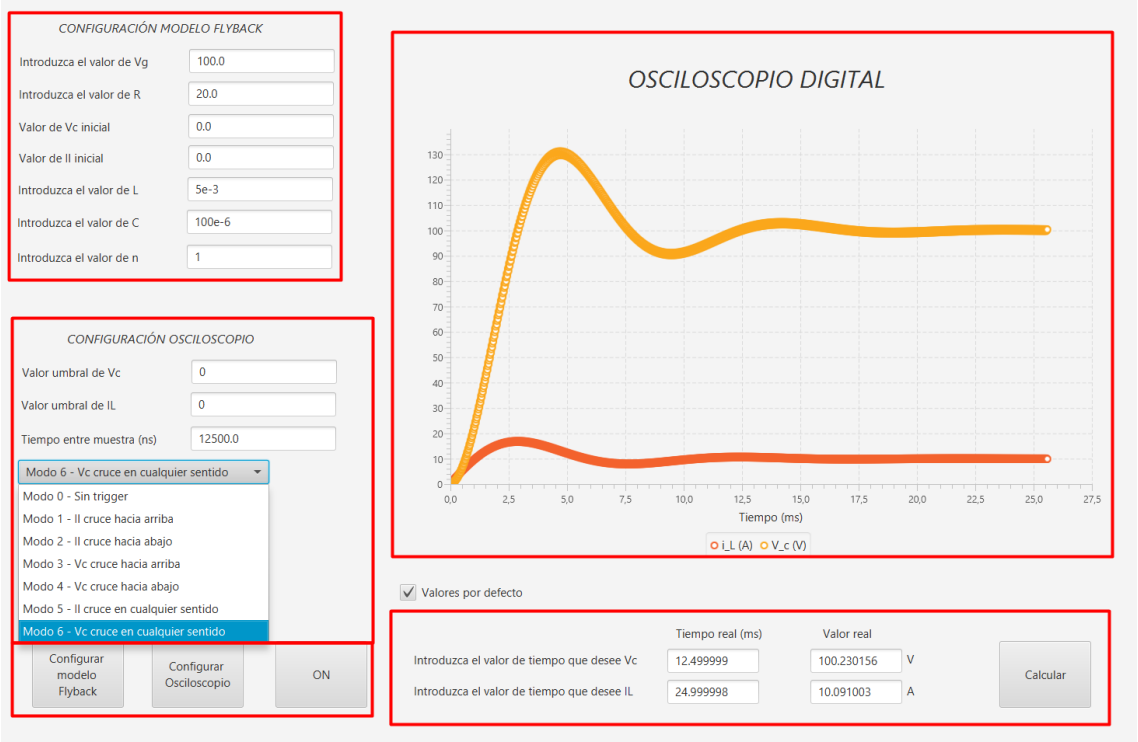

Figura 5-1. Formulario

El formulario de la aplicación es el mostrado en la Figura 5-1. Se puede observar que está formado por cuatro partes que se van a explicar a continuación. La primera de ellas (*CONFIGURACIÓN MODELO FLYBACK*) es la encargada de incluir los valores de configuración del *Modelo Flyback*, en ella se han de introducir todos los valores necesarios de configuración del convertidor. Por defecto, los valores de v_C e i_L inicial serán 0, sin embargo, son configurables. El valor de dT no es configurable, ya que dicho

valor depende de la frecuencia de reloj utilizada en el hardware del sistema. Como para realizar los cálculos se utilizan 16 MHz tal y como se ha explicado en 3.4.4, el valor de *dT* es el inverso, es decir, 0,625 ns.

El segundo bloque (*CONFIGURACIÓN OSCILOSCOPIO*) se corresponde a la configuración del bloque *Oscilo*. En él se puede configurar el modo de realizar la captura, si se desea *trigger* o no, y de qué manera desea el *trigger*; cruce hacia arriba, hacia abajo o en cualquier sentido; el tiempo entre muestra y, además, la variable de estado sobre la que se desea realizar el *trigger*. Como en un osciloscopio tradicional, cuanto mayor sea el tiempo entre muestra, se podrá observar el comportamiento del convertidor durante más tiempo y distinguir mejor la forma de la onda. Sin embargo, si se utiliza demasiado espacio entre muestras, no se conseguirá observar en detalle los cambios del circuito. Internamente el código dispone de un valor mínimo de tiempo entre muestra ya que no puede ser inferior a la frecuencia de funcionamiento del convertidor.

Una vez se han escrito todos los valores de configuración es el momento de enviar dichos valores. Para ello se dispone de 3 botones, el primero de ellos se utiliza para enviar los valores de configuración del *Modelo Flyback* pasando por el bloque *Configurador*; el segundo de ellos es el encargado de enviar los valores de la captura de los datos y, por último, el tercer botón se corresponde con el LOAD, cuya utilidad ya se ha explicado en secciones anteriores. Una vez el modelo está en funcionamiento, cuando se activa el LOAD el formulario entra en el modo de recepción. Esto quiere decir que está escuchando a que le lleguen los valores de la captura. Cuando la captura ya esté completa, gracias al uso de hilos se activa directamente la gráfica y se muestran los datos capturados. Esto nos lleva al último bloque (*OSCILOSCOPIO DIGITAL*) donde se encuentra la gráfica en la que se van a mostrar las dos variables de estado que se han capturado, donde el eje de abscisas (*X*) se corresponde con el tiempo de la captura (µs) y el eje de coordenadas (*Y*) se corresponde con el valor tanto de v_C (V- voltios) como de i_L (A-amperios).

Para que todos los valores sigan un único formato numérico, que es el IEEE-754, hay que transformar todas las variables que se envían y se reciben. Para ello dependiendo del estado en el que nos encontremos se tiene que actuar de una manera o de otra.

- Para el estado de envío hay que traducir de valor decimal a formato IEEE-754. Para ello hay que transformar el valor decimal en binario, y de ahí transformarlo en un array. A la hora de enviar los primeros valores desde la aplicación a la FPGA nos damos cuenta de que debido a que son lenguajes distintos y en plataformas distintas, C++ (Zynq) y Java (Aplicación ordenador), el array llegaba desordenado debido a discrepancias de formatos Big/Little Endian entre los diferentes elementos de comunicación. La solución a ese problema fue enviar el array desordenado para que llegara al destino ordenado, ya que ordenarlo en la recepción no era posible puesto que al tratarse de cadenas de bytes podrías transformar un número positivo en negativo.
- Para el estado de recepción hay que convertir un array de bits en número decimal. Para ello se procede a seguir el esquema de conversión de IEEE-754 según la Figura 13. Con él se consigue obtener el valor decimal de manera correcta.

```
private static float bytetofloat(byte a, byte b, byte c, byte d)
{

    int sgn, mant, exp;//Declaración del signo, mantissa y exponente
    sgn = (a & 128);//El signo se corresponde con el bit de la posición 31 - el 7 bit en un byte
    if (sgn == 0)
        sgn = 1;//Si vale 0 es positivo
    else
        sgn = -1;//Si vale 1 es negativo
    exp = ((a & 0x7F) << 1 | (b & 0x80) >> 7);//Obtención del exponente
    mant = ( b &0x7F) << 16 | (c & 0xFF ) << 8 | ( d & 0xFF);//Obtención de la mantissa

    if (a == 0x00 & b == 0x00 & c == 0x00 & d == 0x00)//Si todos los bytes son 0, el valor es 0
        return (float) 0.0;
    else
        return (float) (sgn * Math.pow(2, (exp-127)) * (1 + mant/Math.pow(2, 23)));//Obtención del valor en decimal
}
```

Figura 5-2. Conversión de binario a decimal

Por último, se encuentra el botón de *Calcular*, que permite averiguar el valor de las variables de estado en un tiempo determinado. Basta con que el usuario introduzca el tiempo de simulación en el que quiera saber el valor, y el sistema mostrará el tiempo real donde se ha capturado ese valor y además el valor de dicha variable en ese tiempo concreto.

Una vez se conoce cómo se ha construido todo el sistema, tanto hardware como software es el momento de probar el funcionamiento global del sistema y comprobar que sea correcto gracias a las fórmulas propuestas en la sección 3.1. Por tanto, la siguiente sección será la correspondiente a resultados.

6 Resultados

Una vez se han implementado todos los bloques, es el momento de realizar una serie de pruebas con el fin de comprobar si el diseño y la implementación en la FPGA se han realizado correctamente. Se realizarán pruebas software con el modelo del convertidor para ver si se comporta como se espera, mediante la herramienta *Questa Sim*. Posteriormente se harán pruebas de integración, donde se probará el sistema final ejecutándose en la FPGA, donde se realizarán diferentes experimentos variando el ciclo de trabajo y los *triggers* del osciloscopio.

6.1 Resultados de implementación en la FPGA

El primer resultado que se muestra es la cantidad de recursos utilizados en el sistema y la velocidad que llega a alcanzar el sistema una vez se implementa el proyecto en la FPGA Xilinx Arty Z7-20. En los resultados se mostrará tanto la velocidad como los recursos utilizados de todo el proyecto en conjunto.

La herramienta que se ha utilizado es el propio programa de Vivado en su versión 2018.3. Dicha herramienta nos proporciona un informe en el que se indican el número de recursos utilizados como pueden ser el número de LUTs (*Look Up Table*), el número de multiplicadores usados (DSPs), el número de Block RAMs, etc.

Recurso	LUTs	Flip Flops	DSPs	Block RAMs	T_{CLK}
Proyecto completo (%total FPGA)	6996 (13,15%)	7977 (7,50%)	10 (4,55%)	6,00 (4,29%)	100 MHz
Periférico (%total FPGA)	2979 (5,60%)	2425 (2,28%)	10 (4,55%)	4.00 (2,86%)	100/16 MHz

Tabla 6-1. Uso de recursos del sistema

Como se puede observar en la Tabla 6-1 el número de DSPs total del proyecto es el número correspondiente al periférico, que es donde se realizan las operaciones necesarias para obtener las variables de estado. El número de Flip Flops es ligeramente elevado, debido a que como hay varias variables de estado (i_L y v_c), se sincronizan varios datos entre varios dominios de reloj, creando para ello dobles registros, y, por tanto, creando registros intermedios para separar las diferentes etapas de captura de datos, adaptación para la escritura en Block RAMs, envío por la interfaz AXI STREAM, etc. Aun así, se puede observar cómo apenas suponen el 8% del total de Flip Flops que se pueden usar en el sistema, y sólo el 2,28% se corresponden al periférico. Por otra parte, se observa como el número de Block RAMs también es bajo, el 4,29%. El recurso que más se utilizan son las LUTs, que se usa un poco más del 13% del total. En total se observa que el uso de recursos total del sistema es bastante bajo, teniendo en cuenta que es una FPGA de bajo coste (190€ aproximadamente), en contraposición con las FPGAs de alto coste que utilizan simuladores HIL comerciales, las cuales llegan a costar miles de euros.

Se ha intentado maximizar la velocidad del sistema en sus diferentes partes. El modelo del convertidor tiene operaciones aritméticas complejas que no pueden ser mejoradas mediante pipelining haciendo posible únicamente una frecuencia de 16 MHz.

Sin embargo, en esta aplicación, el pipeline no es buena idea porque el modelo está realimentado ya que se necesita el valor anterior para calcular uno de nuevo. De esa forma, tanto la latencia como el rendimiento del modelo no se verían mejorados. Con esa frecuencia de 16 MHz se consigue cumplir con los requisitos de tiempo proporcionados por el programa. Sin embargo, el resto de los componentes del sistema pueden ejecutarse a 100 MHz. Por tanto, se puede observar que el periférico usa dos relojes, 16 MHz para el *Modelo Flyback*, y 100 MHz para lo demás componentes del circuito.

6.2 Simulación del convertidor

Para realizar el circuito en coma flotante primero se realizó un convertidor con variables de tipo *real* tal y como se indicó en la sección 3.2. Para comprobar el funcionamiento correcto del convertidor de potencia se realizan dos simulaciones, una utilizando las variables de tipo *real*, y otra con variables en formato float. Siempre y cuando ambos circuitos se encuentren en condiciones iguales ambas simulaciones tienen que ser idénticas. Para el modelo de referencia real se debe usar la misma frecuencia de reloj que para el modelo en coma flotante, es decir, 16 MHz, además el tiempo de muestra deberá ser nulo para que ambos modelos sigan el mismo esquema.

Al realizar la simulación se observa que el valor de v_C está a cero durante "un largo período de tiempo" , comportamiento normal del Flyback cuando parte de reposo. Esto es debido a que hasta que el transformador no se cargue de energía con el interruptor cerrado, para luego liberar dicha energía a la parte secundaria del circuito, el valor de v_C permanecerá a cero.

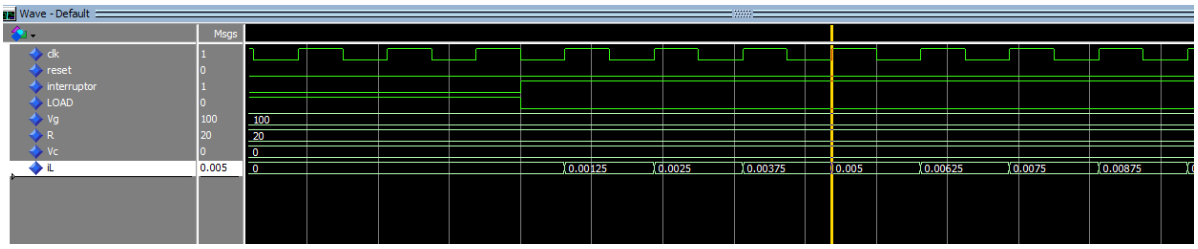

Figura 6-1. Simulación Questa Sim convertidor real primeros valores i_L

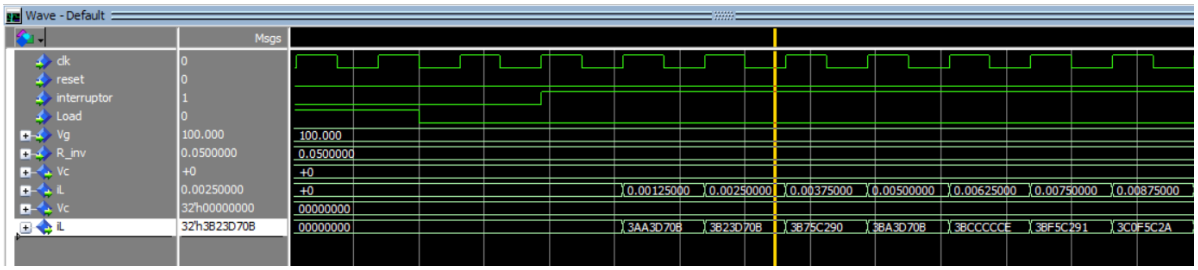

Figura 6-2. Simulación Questa Sim convertidor real primeros valores i_L

Como se puede comprobar en las Figuras 6-1 y 6-2, el comportamiento de ambos circuitos es el mismo, esto nos hace indicar en un principio que el circuito se ha desarrollado correctamente. Pese a que las variables de tipo *real* poseen una mayor precisión, en este caso no se observan diferencias entre los valores puesto que no se trata de valores con demasiados decimales. Para confirmar realmente que el convertidor con

variables tipo float está funcionando correctamente se han de comprobar también los valores del voltaje de salida.

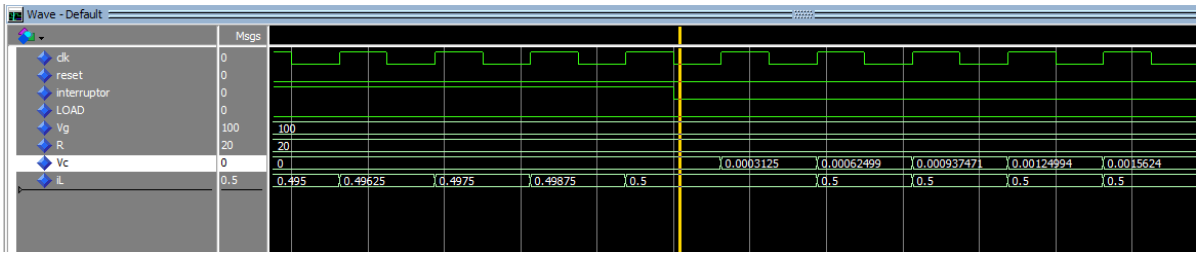

Figura 6-3. Simulación Questa Sim convertidor real primeros valores v_C

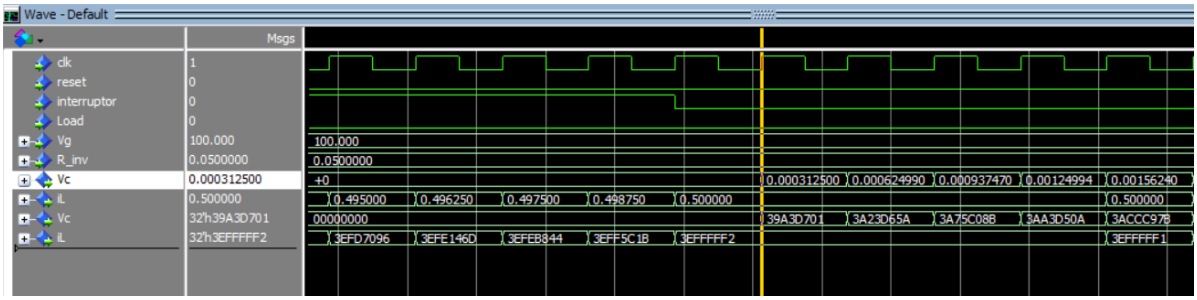

Figura 6-4. Simulación Questa Sim convertidor float primeros valores v_C

En las Figuras 6-3 y 6-4 se comprueba como el modelo creado con variables tipo float es idéntico al tipo *real*, debido a que su comportamiento es igual. De hecho se observa como el valor de i_L se mantiene constante durante unos ciclos en ambas simulaciones. También se observa como la energía del transformador se libera por primera vez a la parte secundaria del circuito, haciendo que la tensión de salida empiece a aumentar. Por último, para comprobar el correcto funcionamiento, se simula una cantidad de tiempo necesaria para ver el comportamiento final del circuito. Mediante la herramienta de Questa Sim se puede observar en forma de onda el comportamiento del convertidor. Los valores de configuración del convertidor Flyback serían los siguientes:

Variable	*Valor*	*Variable*	*Valor*
V_G	100,0 V	***R (R_inv=1/R)***	20,0 Ω (0,05 S)
L	5e-3 H	***C***	100e-6 C
n	1	***dT***	6.25e-8 s
F_{sw}	16 MHz	$F_{interruptor}$	20 kHz

Tabla 6-2. Valores de configuración del convertidor Flyback para la simulación software

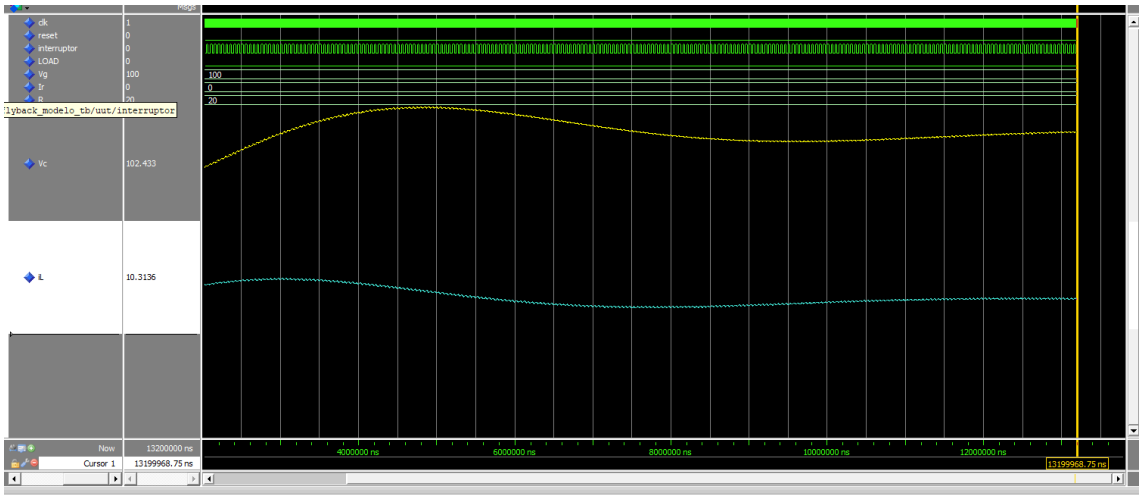

Figura 6-5. Simulación Questa Sim convertidor *real*

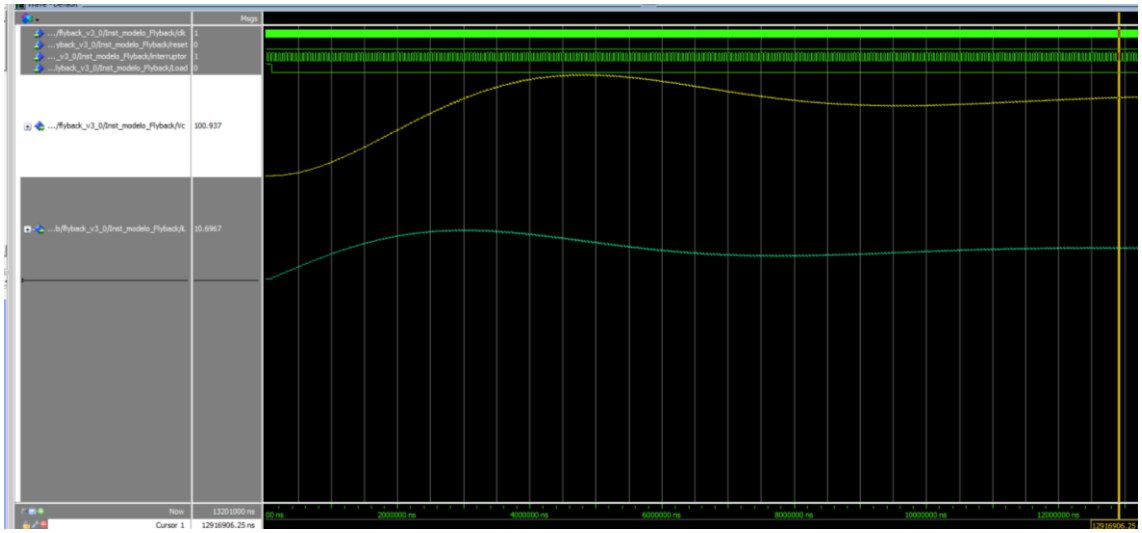

Figura 6-6. Simulación Questa Sim convertidor float

Finalmente en las Figuras 6-5 y 6-6 se confirma el correcto funcionamiento del circuito tanto para el modelo con variables de tipo *real*, como el creado en float. Esto se debe a que el valor final que adquiere el voltaje de salida está en torno a los 100 V. Debido a que el ciclo de trabajo es igual al 50% es el valor esperado siempre que la tensión de entrada sea de 100 V, que se corresponde con el valor introducido como se muestra en la Tabla 6-2. Las futuras simulaciones del proyecto completo seguirán dicho esquema siempre y cuando el ciclo de trabajo sea igual al 50%, que es la condición que se sigue en esta simulación.

6.3 Comprobación de valores mediante el ILA

Gracias al componente System ILA (*Integrated Logic Analyzer*) es posible comprobar si los valores que se están mandando a la FPGA o saliendo de ella son los esperados. Para ello, basta con introducir las señales que se desean visualizar en el componente e introducir una serie de condiciones que han de cumplirse para que salte el *trigger*. Por tanto, se van a probar dos situaciones, la primera de ellas una trama de

configuración del modelo matemático del convertidor, es decir, la visualización del protocolo AXI-Lite, mientras que la segunda será una trama de envío desde la FPGA hacia la aplicación, es decir, una visualización del protocolo AXI-Stream.

Primero se introducen los valores de configuración con los que se va a probar el modelo del convertidor:

Variable	*Valor*	*Variable*	*Valor*
V_G	100,0 V	R	20,0 Ω
$v_{C\_inicial}$	0,0 V	$i_{L\ inicial}$	0,0 V
L	5e-3 H	C	100e-6 C
n	1	dT	6,25e-8 s

Tabla 6-3. Valores del convertidor Flyback

Con los valores introducidos en la Tabla 6-3, se conocen los valores que se van a enviar desde la aplicación a la FPGA, pasando por el protocolo AXI-Lite. Puesto que los cálculos se realizan utilizando el protocolo IEEE-754, se deben comprobar dichos valores. Por tanto, los valores que se van a enviar ordenados siguiendo el orden descrito en la sección 5.1 son:

Variable	Valor decimal	Valor IEEE-754	Valor recibido en el AXI-Lite
Trama de control	-	0x3F800001	No se envía al AXI-Lite
V_G	100,0	0x42C80000	0x42C80000
$^1/_R$	0,05	0x3D4CCCCD	0x3D4CCCCD
$v_{C\_inicial}$	0,0	0x00000000	0x00000000
i_{L_incial}	0,0	0x00000000	0x00000000
$^{dT}/_C$	6,25e-4	0x3751B718	0x3751B718
$^{dT}/_L$	1,25e-5	0x3A23D70A	0x3A23D70A
$^1/_n$	1	0x3F800000	0x3F800000

Tabla 6-4. Valores de configuración del convertidor

En la Tabla 6-4 se muestran los valores que se envían desde la aplicación y su valor en formato IEEE-754. Como se puede observar, en primer lugar se envía un valor de control para saber si se trata de una trama de configuración del convertidor, del Oscilo, o si se trata de la variable LOAD. Dicho valor únicamente se leerá en el software de la FPGA pero no se enviará al circuito. Una vez se conocen los valores, es el momento de comprobar lo que le llega a la FPGA mediante el analizador lógico.

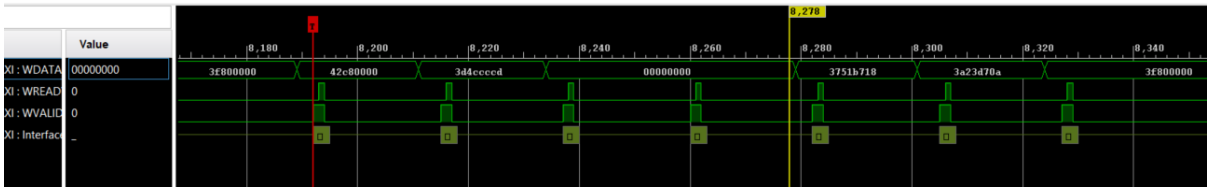

Figura 6-7. Simulación ILA trama de configuración

Como se puede observar en la Figura 6-7, los valores se reciben por orden en el AXI-Lite, tal y como se ha indicado en secciones anteriores. Con dicha imagen se comprueba que el protocolo de comunicación se ha realizado correctamente. Por tanto es el momento de visualizar el funcionamiento del protocolo AXI-Stream.

La forma de comprobación de estos valores será mediante la simulación en Questa Sim, puesto que dichos valores deberían ser los que los mostrados en la simulación ILA. Para ello se van a escoger determinados valores que serán los siguientes:

- Los 4 primeros valores de la memoria del oscilo correspondientes a i_L.
- Los 2 últimos valores de la memoria del oscilo de i_L y los 4 primeros de v_C.
- Los 2 últimos valores de la memoria del oscilo correspondientes a v_C.

Estos valores son los que delimitan el principio y fin presentes en la memoria del oscilo por cada variable de estado que se está analizando. Ambas simulaciones van a tener los mismos valores tanto en la configuración del modelo matemático del convertidor como en la captura de los datos. Los valores de configuración son los presentados anteriormente y las condiciones de captura será cuando salte un *trigger* superior a 10 V para la variable de estado v_C. En primer lugar se presenta la simulación en el programa Questa Sim con los primeros valores de i_L tanto en decimal como en hexadecimal en el formato IEEE-754.

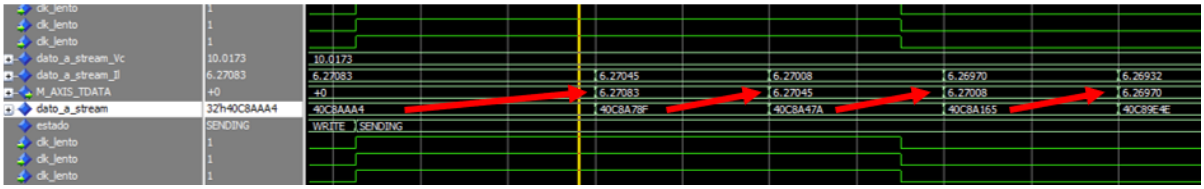

Figura 6-8. Simulación Questa Sim convertidor Flyback valores i_L

Como se puede observar en la Figura 6-8, el valor se entrega al protocolo de envío con un ciclo de retraso con respecto a la obtención de dicho valor. Una vez tenemos los valores decimales y hexadecimales se incluye la captura en el ILA.

Figura 6-9. Simulación ILA del proyecto mediante ILA valores i_L

Más adelante se añadirán todos los valores correspondientes a las capturas tanto en decimal como en hexadecimal para poder realizar una mayor comparación. Sin embargo, comparando solamente los valores hexadecimales se observa que los valores no son iguales pero son bastante similares. Esto puede deberse a que al ser el interruptor una señal externa al sistema puede haber pequeñas variaciones en la lectura provocando pequeñas variaciones que provocarán que los valores no sean idénticos. En las tablas de comparación se observará que los valores son muy similares. Además en la Figura 6-9 se puede comprobar como los datos se envían siempre y cuando las variables TVALID y TREADY estén ambas activadas a la vez. Para observar cómo funciona el envío de toda la trama completa se va a mostrar una captura acerca del cambio de i_L a v_C.

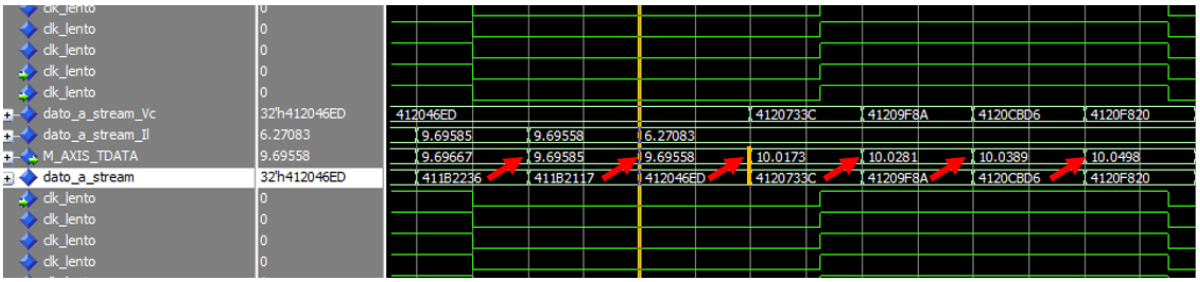

Figura 6-10. Simulación Questa Sim Flyback valores i_L e v_C

Como se puede observar en la Figura 6-10 una vez se envían los 2048 valores correspondientes a i_L se empiezan a enviar automáticamente los valores correspondientes a v_C. Para comprobar también el comportamiento y valores en una simulación a tiempo real se incluye la captura del analizador ILA.

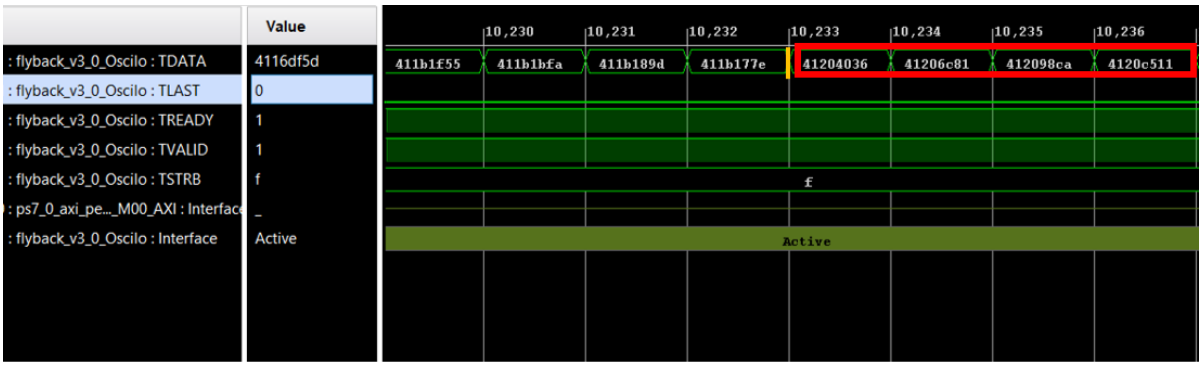

Figura 6-11. Simulación ILA del cambio de variable, valores i_L e v_C

En la Figura 6-11 se puede comprobar el mismo comportamiento que en la simulación software. En cuanto una variable de estado se termina de enviar automáticamente empieza la otra. Una vez se ha presentado el funcionamiento del circuito se comparan los valores entre ambas simulaciones.

Valor i_L	Questa Sim Hexadecimal	Questa Sim Decimal	System ILA	Aplicación ordenador
1	0x40C8AAA4	6,27083	0x40C89664	6,268358
2	0x40C8A78F	6,27045	0x40C8934F	6,267982
3	0x40C8A47A	6,27008	0x40C8903A	6,267605
4	0x40C8A165	6,26970	0x40C88D25	6,267229
2047	0X411B2236	9,69585	0x411B189D	9,693509
2048	0x411B2117	9,69558	0x411B177E	9,693235

Tabla 6-5. Resultados de la variable de estado i_L

Valor v_C	Questa Sim Hexadecimal	Questa Sim Decimal	System ILA	Aplicación ordenador
1	0x412046ED	10,0173	0x41204036	10,015676
2	0x4120733C	10,0281	0x41206C81	10,026490
3	0x41209F8A	10,0389	0x412098CA	10,037302
4	0x4120CBD6	10,0498	0x4120C511	10,048112
2047	0x41AF1E16	21,8897	0x41AF166A	21,885946
2048	0x41AF2918	21,8951	0x41AF216C	21,891321

Tabla 6-6. Resultados de la variable de estado v_C

Como se ha dicho anteriormente los valores no son idénticos a los que cabría esperar de una simulación con los mismos valores de configuración. Tal y como se ha introducido anteriormente estas pequeñas variaciones pueden originarse en la lectura del interruptor, ya que es una señal externa al circuito. Sin embargo, se puede comprobar como los valores si son muy similares. Otro de los factores puede ser el factor de redondeo a la hora de realizar la transformación de los números del formato IEEE-754 a decimal.

6.4 Comprobación convertidor Flyback

Por último, se va a analizar si nuestro proyecto cumple con todas las fórmulas incluidas en secciones anteriores. Esto se realiza mediante varias simulaciones. La primera de ellas será el comportamiento de la corriente en régimen permanente. Como se explicó en la sección 3.1 el comportamiento de la corriente en régimen permanente tiene que ser el siguiente:

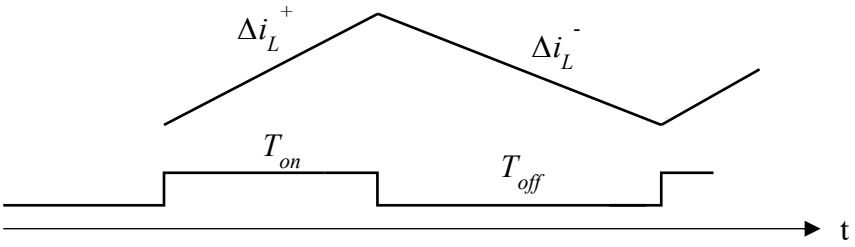

Figura 6-12. Comportamiento i_L en régimen permanente

Con lo que se obtiene que:

$$\Delta i_L^+ + \Delta i_L^- = 0 \qquad (6.1)$$

Como se indica en la fórmula 6.1, el ciclo de trabajo no tiene efecto sobre el incremento de i_L positivo y negativo en régimen permanente en un ciclo del interruptor. Dicha ecuación se comprobará mediante una captura del osciloscopio digital. Para realizar dicha comprobación se realizarán pruebas con duty = 25%, 50% y 75%:

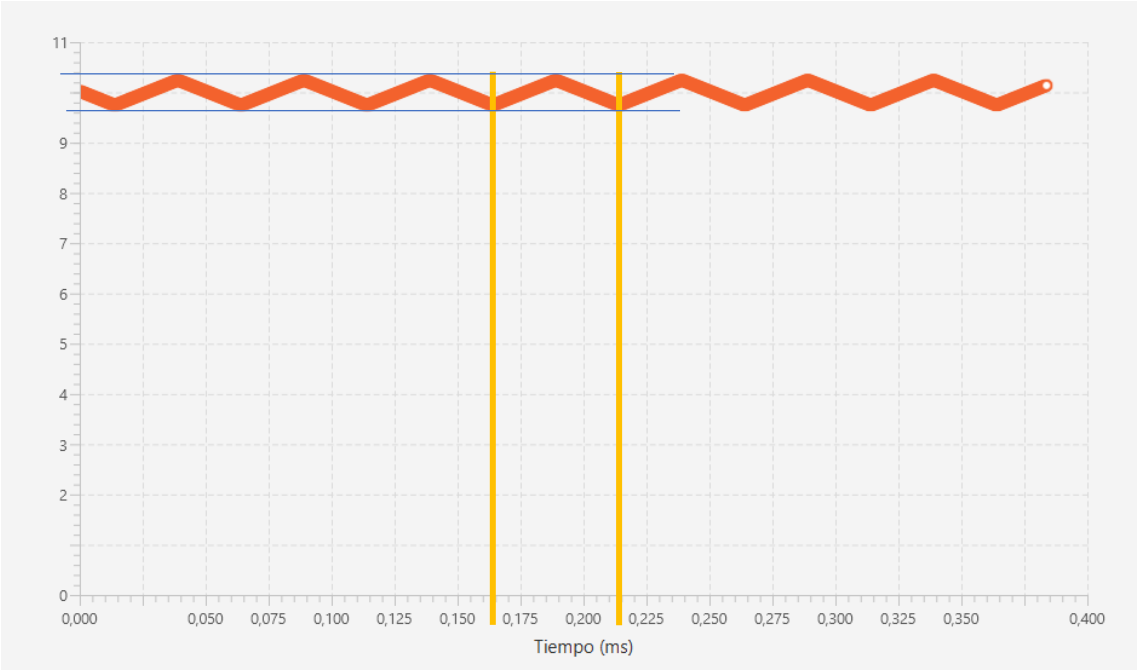

Figura 6-13. Captura i_L con duty cicle = 50%

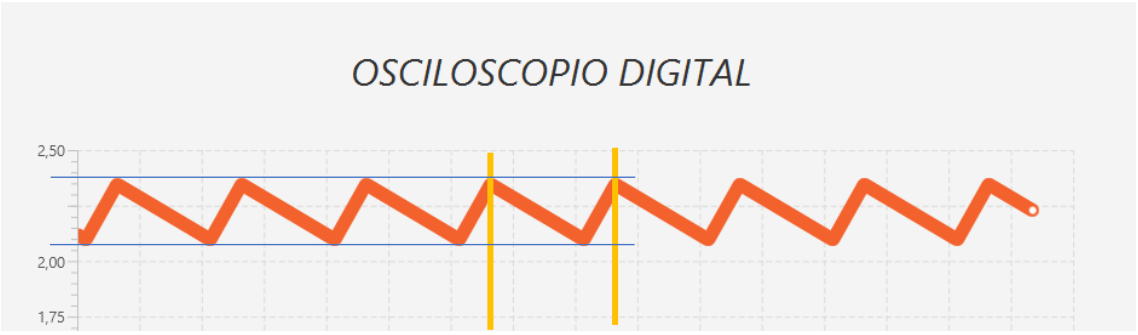

Figura 6-14. Captura i_L con duty cicle = 25%

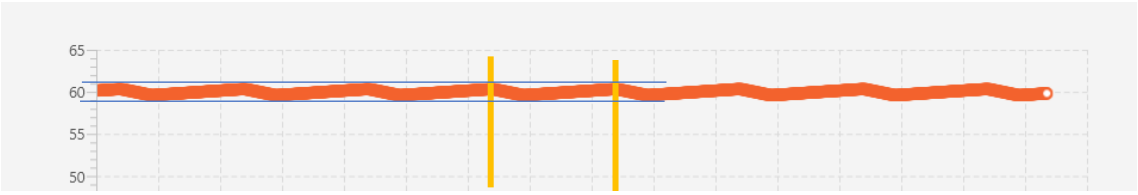

Figura 6-15. Captura i_L con duty cicle = 75%

Como se puede comprobar con las Figuras 6-13, 6-14 y 6-15, cuanto mayor sea el valor del ciclo de trabajo, más tiempo dura la pendiente positiva, provocando a su vez que la pendiente de dicho incremento sea menor. Además, esto provoca que el tiempo de decremento sea inferior, provocando que su pendiente sea mayor para conseguir bajar la misma cantidad que el incremento superior pero en una cantidad de tiempo inferior. Como se puede observar mediante las líneas azules, los valores de las oscilaciones de la variable de estado son siempre los mismos. Además, como la frecuencia del interruptor es la misma para las tres simulaciones, los periodos de los triángulos que se crean son el mismo.

Por tanto, ahora se va a comprobar la segunda fórmula que estamos analizando, la cual corresponde con el voltaje de salida, que en este proyecto se corresponde con v_C. La fórmula que se va a comprobar es la siguiente:

$$V_G = \frac{v_{out}}{n} * \frac{1-d}{d} \rightarrow v_{out} = V_G * \frac{n*d}{1-d} \qquad (6.2)$$

Vista la fórmula se va a comprobar para cinco ciclos de trabajo distintos, los dos extremos, 0 y 1, es decir, un duty cicle del 0% y del 100%. Una vez se tengan dichos valores se comprobarán los valores intermedios, es decir, 25%, 50% y 75% con el fin de observar como el convertidor puede realizar la misma acción que los circuitos *Buck* y *Boost*. Las condiciones del convertidor y captura serán las mismas para todas las capturas. En la parte de configuración serán los mismos valores que la Tabla 6-2, mientras que en la captura se intentará observar el comportamiento del circuito en un periodo más largo de tiempo. Por tanto, se va a simular un total de 25,6 ms ya que el tiempo entre muestra será de 12500 ns.

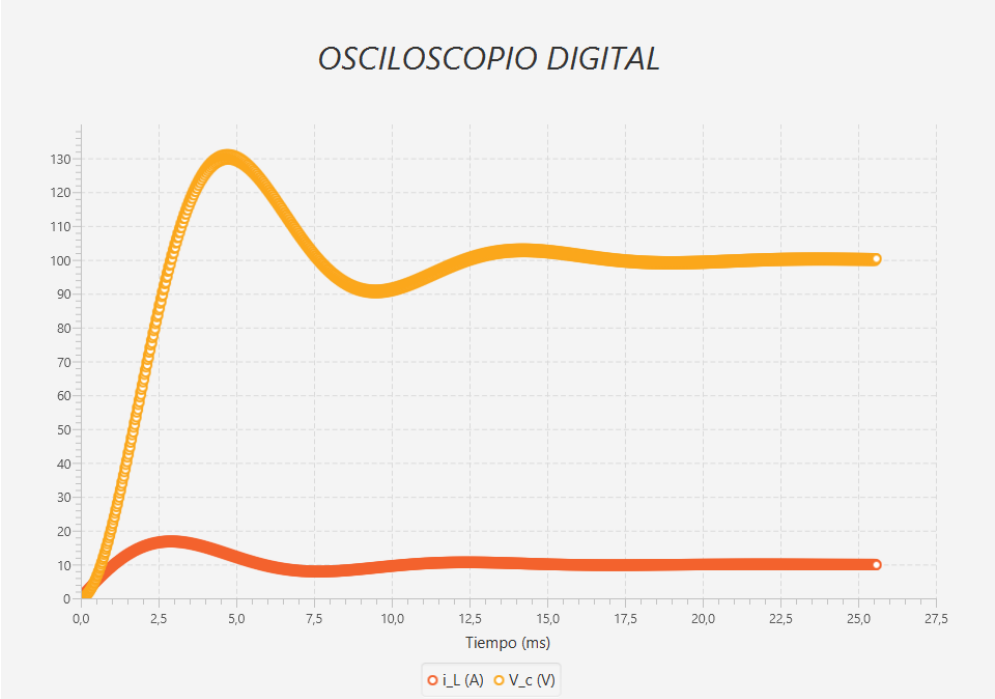

Figura 6-16. Simulación ciclo de trabajo = 50%

Como se puede observar en la Figura 6-16, el sistema sobreoscila para finalmente conseguir estar estable en 100 V, que es el valor proporcionado por la fuente de alimentación. Puesto que el ciclo de trabajo = 50%, el valor de la tensión de salida v_C = V_G*n = 100*1 = 100 V. Por tanto, se comprueba que el funcionamiento es correcto. Para comprobarlo de manera numérica se van a seleccionar unos valores aleatorios al final de la captura para confirmar el correcto funcionamiento. Para ello basta con introducir el valor de tiempo de simulación que se desee visualizar y pulsar el botón de *Calcular*.

Tiempo de simulación	Valor de v_C	Valor de i_L
21,25	99,573975	10,1304655
22,499998	99,92249	10,136398
23,75	100,01897	10,112575
24,999998	99,92106	10,0885
25,5875	100,32854	9,882896

Tabla 6-7. Valores de las variables de estado

Mediante la Tabla 6-7 se confirma como el valor final de la simulación se encuentra en torno al valor esperado, que son los 100 V. Por tanto, mediante todas las simulaciones realizadas se confirma que el modelo matemático del convertidor se ha realizado correctamente.

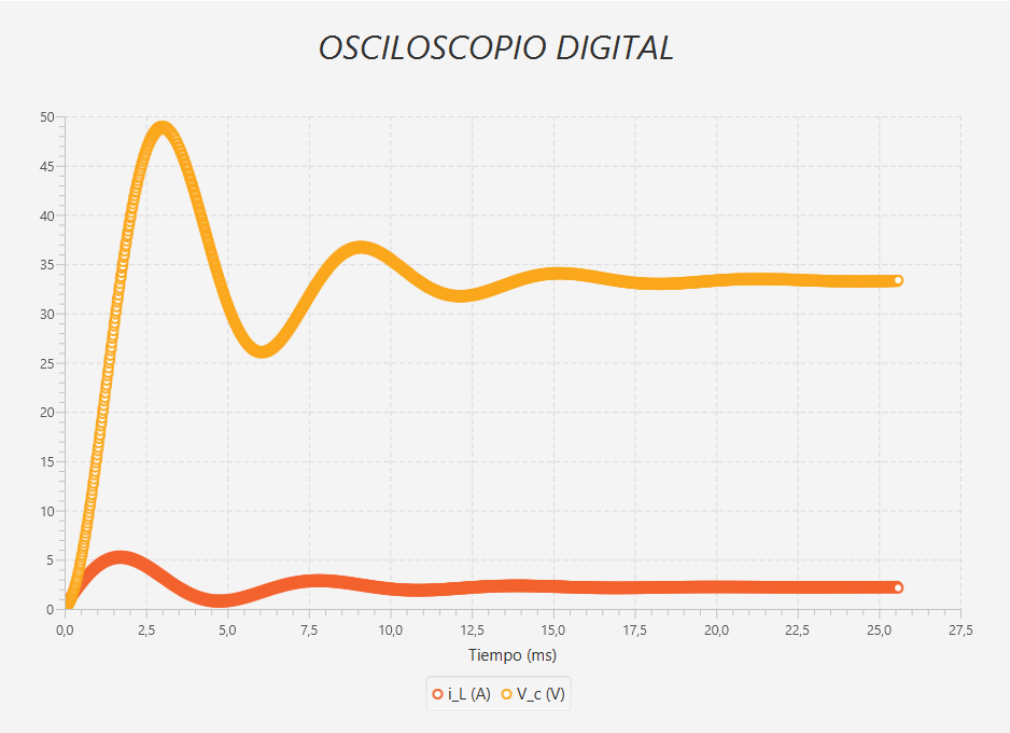

Figura 6-17. Simulación ciclo de trabajo = 25%

Como se puede observar en la Figura 6-17, también se cumple la fórmula planteada, en esta ocasión el rango del ciclo de trabajo se encuentra en el intervalo equivalente a un circuito reductor o *Buck*. Es por ello que el voltaje de salida no llega a alcanzar su valor propuesto, sino que se reduce un 33%. Si resolvemos la fórmula planteada:

$$v_{out} = V_G * \frac{n * d}{1 - d} = V_G * \frac{0{,}25}{1 - 0{,}25} = 100 * \frac{1}{3} = 33{,}33\ V$$

Se comprueba el correcto funcionamiento del circuito y el cumplimiento de las fórmulas. Para el caso en el que el ciclo de trabajo = 75% el valor será equivalente a 3 veces el valor de la fuente, es decir, 300 V:

$$v_{out} = V_G * \frac{n * d}{1 - d} = V_G * \frac{0{,}75}{1 - 0{,}25} = 100 * \frac{3}{1} = 300\ V$$

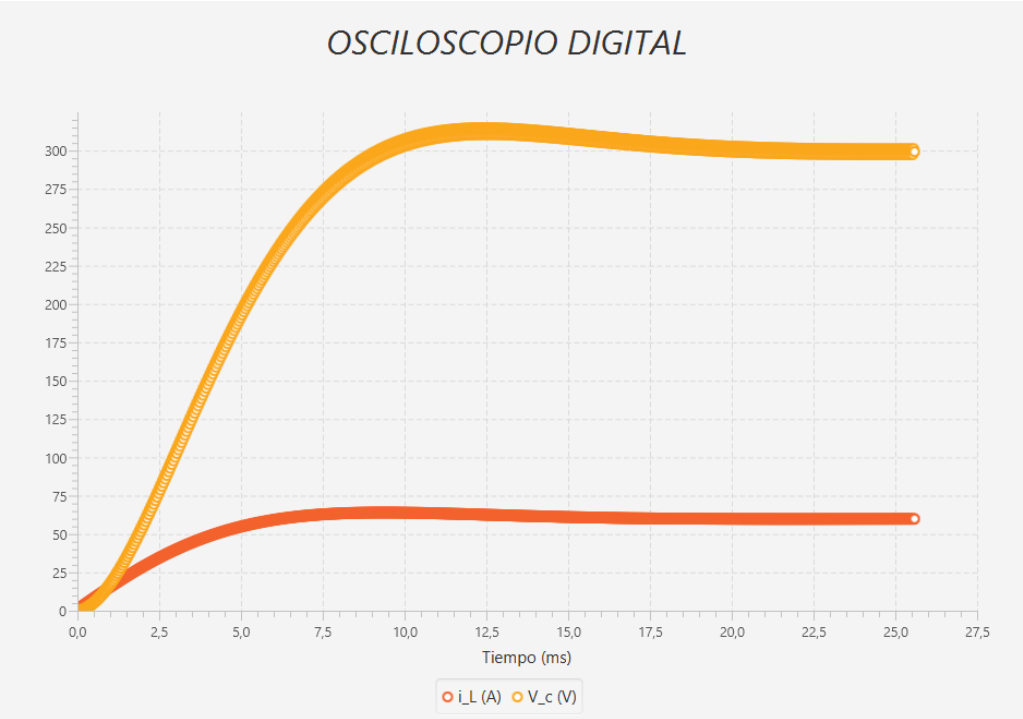

Figura 6-18. Simulación ciclo de trabajo = 75%

En la Figura 6-18 se pueden observar esos 300 V de voltaje de salida. Además en esta simulación se observa que la sobre oscilación del circuito es mínima debido a que el valor del voltaje de salida es mucho mayor que el proporcionado por la fuente. Debido a esto, se puede comprobar como cuando el intervalo del duty cicle se encuentra entre (0.5-1) el circuito se comporta como si fuera un elevador o *Boost*. Por tanto, vistos los valores intermedios únicamente quedan los extremos, es decir, cuando el ciclo de trabajo es 0% o 100%. Como se dijo en la ecuación 6.2, el valor del voltaje de salida será de 0 e ∞ respectivamente.

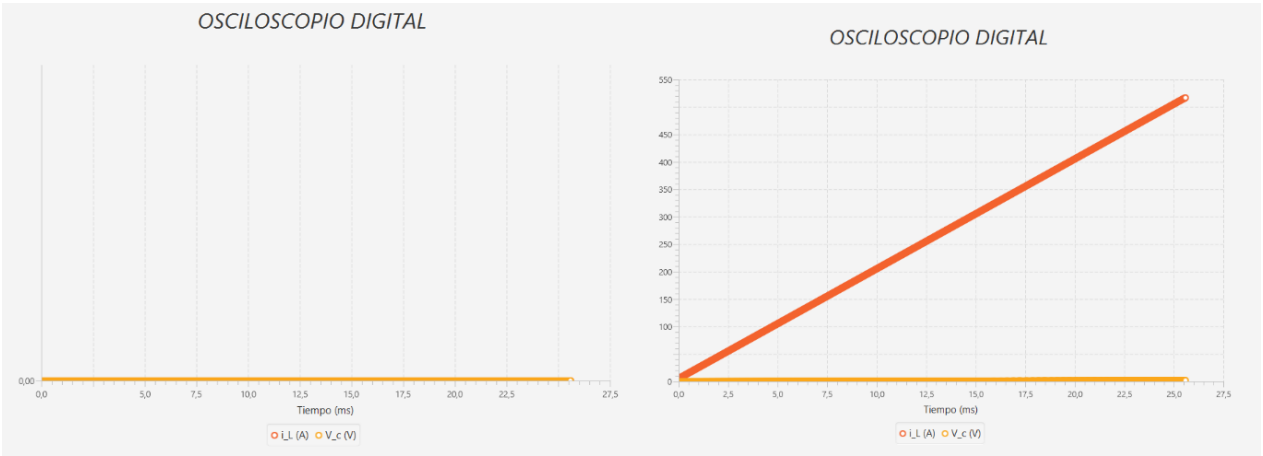

Figura 6-19. Simulación ciclo de trabajo = 0% y 100%

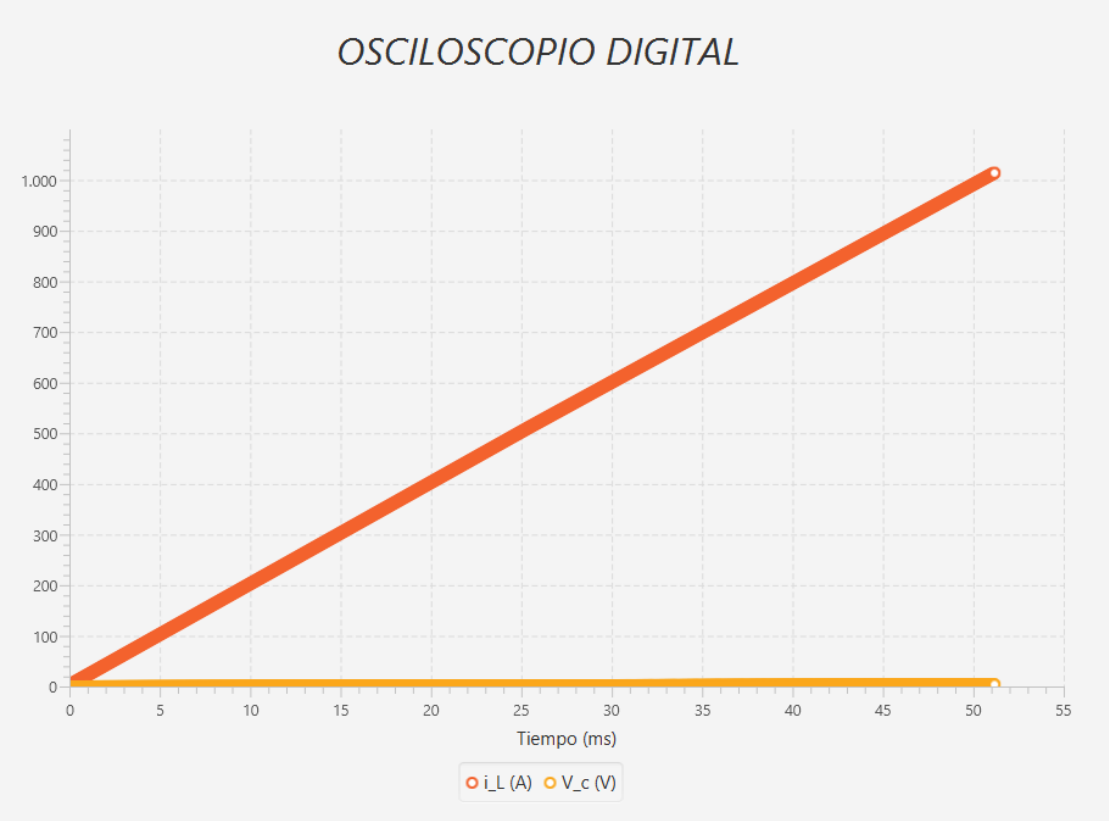

Figura 6-20. Simulación ciclo de trabajo = 100%

En la Figura 6-19 se puede observar la gran diferencia entre los dos extremos, mientras que para un ciclo de trabajo del 0% el voltaje de salida siempre será 0, para un valor de 100% el voltaje no para de aumentar tal y como se demuestra sustituyendo en las fórmulas propuestas:

$$v_{out} = v_G * \frac{n * d}{1 - d} = v_G * \frac{n * 0}{1 - 0} = 100 * \frac{0}{1} = 0\ V$$

$$v_{out} = v_G * \frac{n * d}{1 - d} = v_G * \frac{n * 1}{1 - 1} = 100 * \frac{n}{0} = \infty$$

En la Figura 6-20 se ha configurado la captura con un tiempo entre muestra mayor para observar que ese voltaje continúa creciendo, llegando, por ejemplo, a 1000 V en poco más de 51 ms.

6.5 Funcionamiento del trigger

Debido a que la idea de este proyecto es realizar un osciloscopio digital, éste debe contener una función correspondiente al *trigger* para poder observar mejor la forma de la onda. Debido a que se han propuesto seis tipos de *trigger* distintos, se van a hacer pruebas de tres de ellos, una con la variable de estado v_C y dos con i_L.

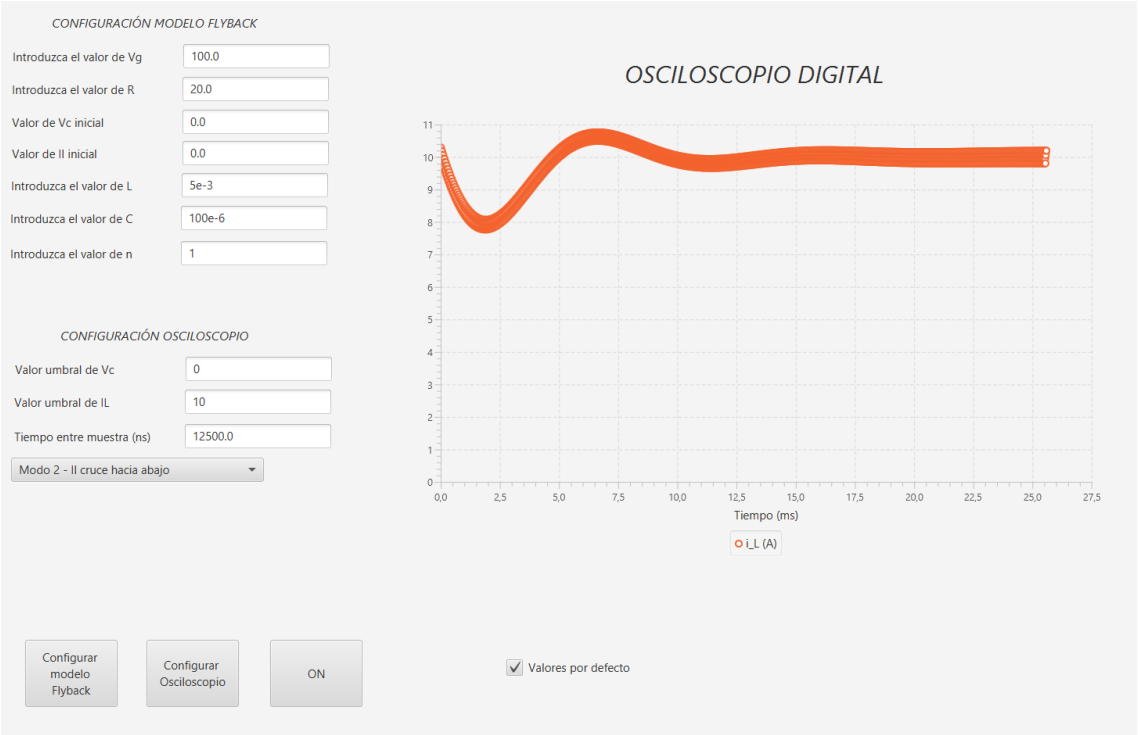

Figura 6-21. Simulación con modo *trigger* - i_L cruce hacia abajo

En la Figura 6-21 se puede observar cómo el sistema empieza a capturar cuando la intensidad se encuentra bajando en 10 V. Si se compara con la Figura 29, se puede observar la diferencia de la onda entre capturar a partir de 0 A, o capturar a partir de 10 A.

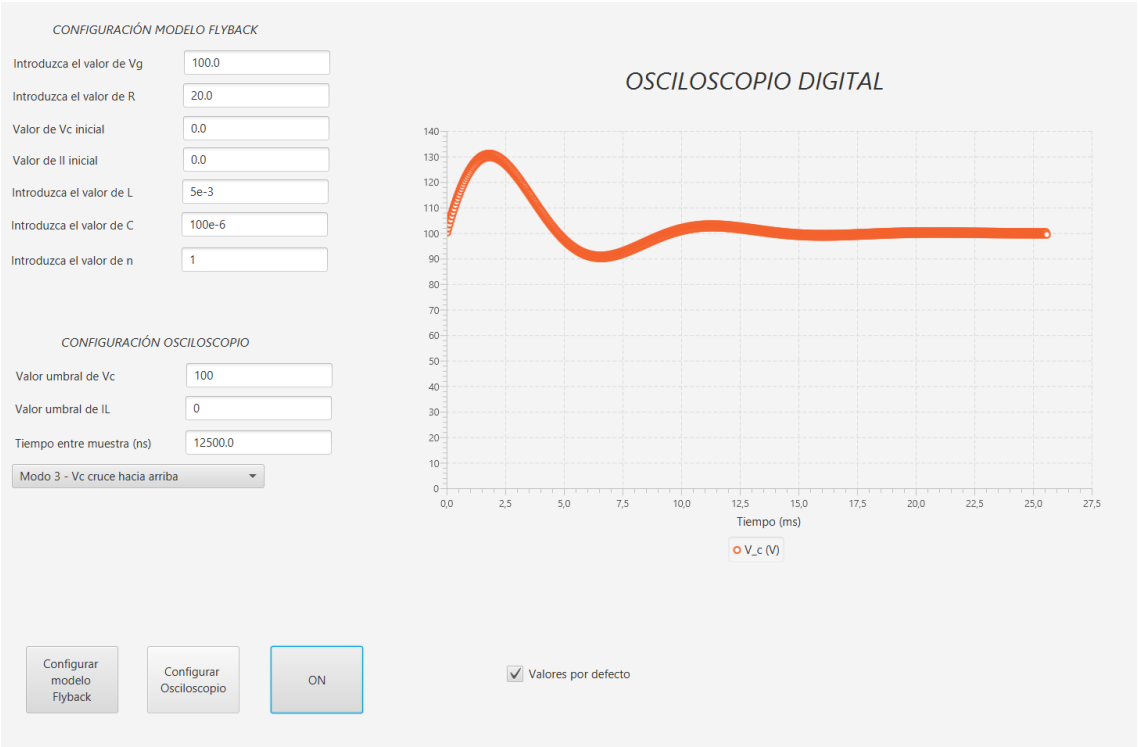

Figura 6-22. Simulación con modo *trigger* - v_C cruce hacia arriba

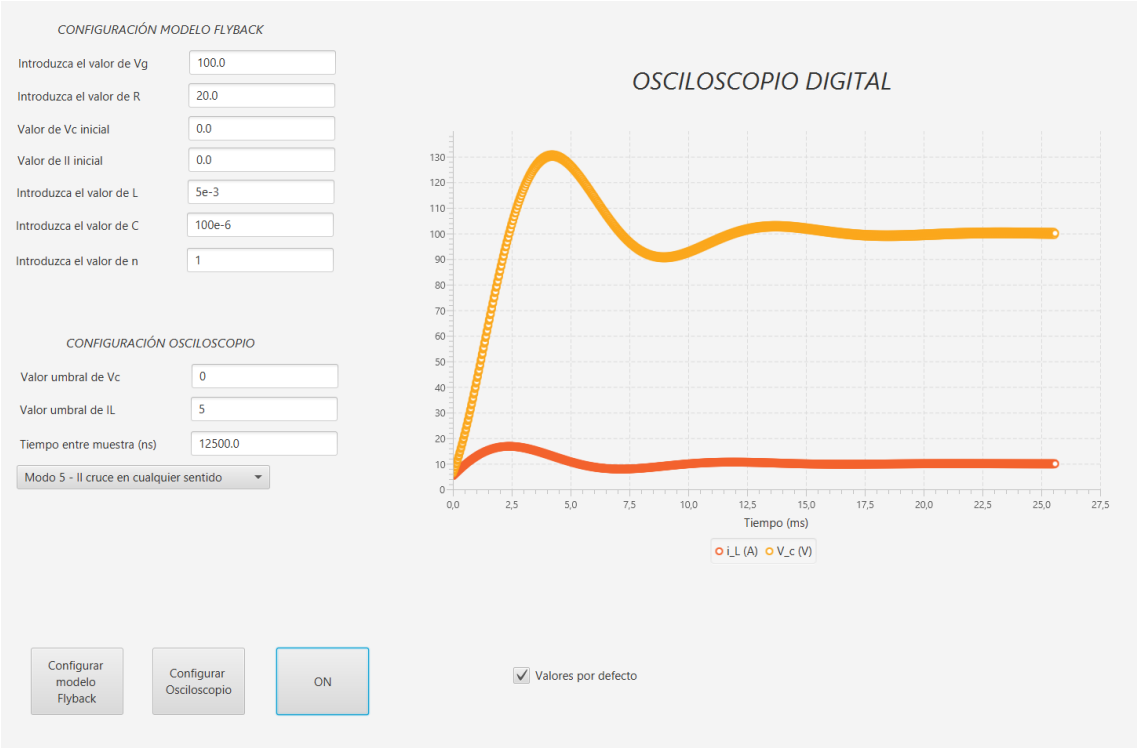

Figura 6-23. Simulación con modo *trigger* - i_L cruce en cualquier sentido

En las Figuras 6-21, 6-22, 6-23 se puede comprobar como el sistema empieza a capturar cuando el valor de la variable de estado se cruza tanto para arriba como para abajo con los valores umbrales de éstas. Tanto en el cruce hacia arriba como hacia abajo pueden existir ocasiones en las que el *trigger* salte en un punto distinto al que se espera. Esto es debido al rizado que posee el circuito proporcionado por la frecuencia del interruptor. Para frecuencias de interruptor menores, mayor será el rizado; por el contrario, cuanto mayor sea la frecuencia del interruptor, el rizado será menor. Sin embargo, se observa un correcto funcionamiento del sistema de captura.

Para comprobar el efecto que tiene el interruptor en el rizado del sistema se van a incluir una serie de simulaciones con distintas frecuencias de interruptor para observar la diferencia. En todas las simulaciones anteriores, la frecuencia del interruptor es de 20 kHz, la cual se ha generado mediante un generador de ondas.

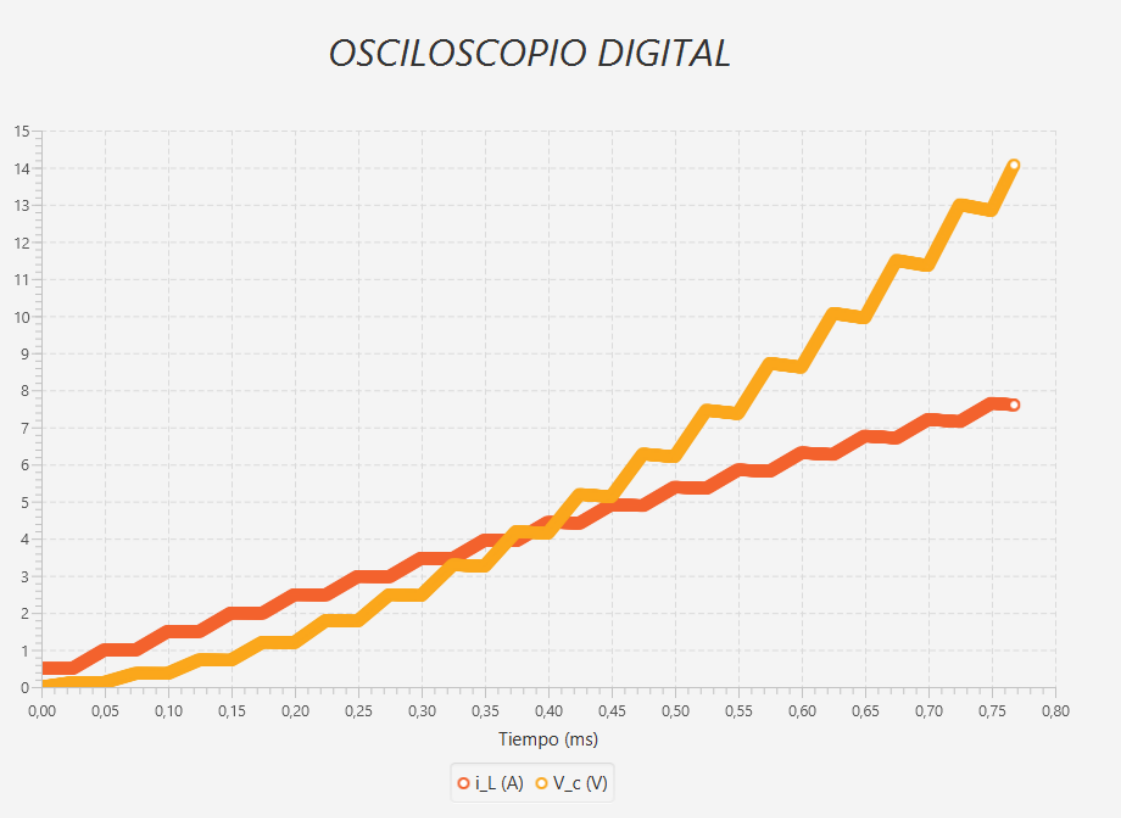

Figura 6-24. Simulación hardware con frecuencia del interruptor = 20 kHz

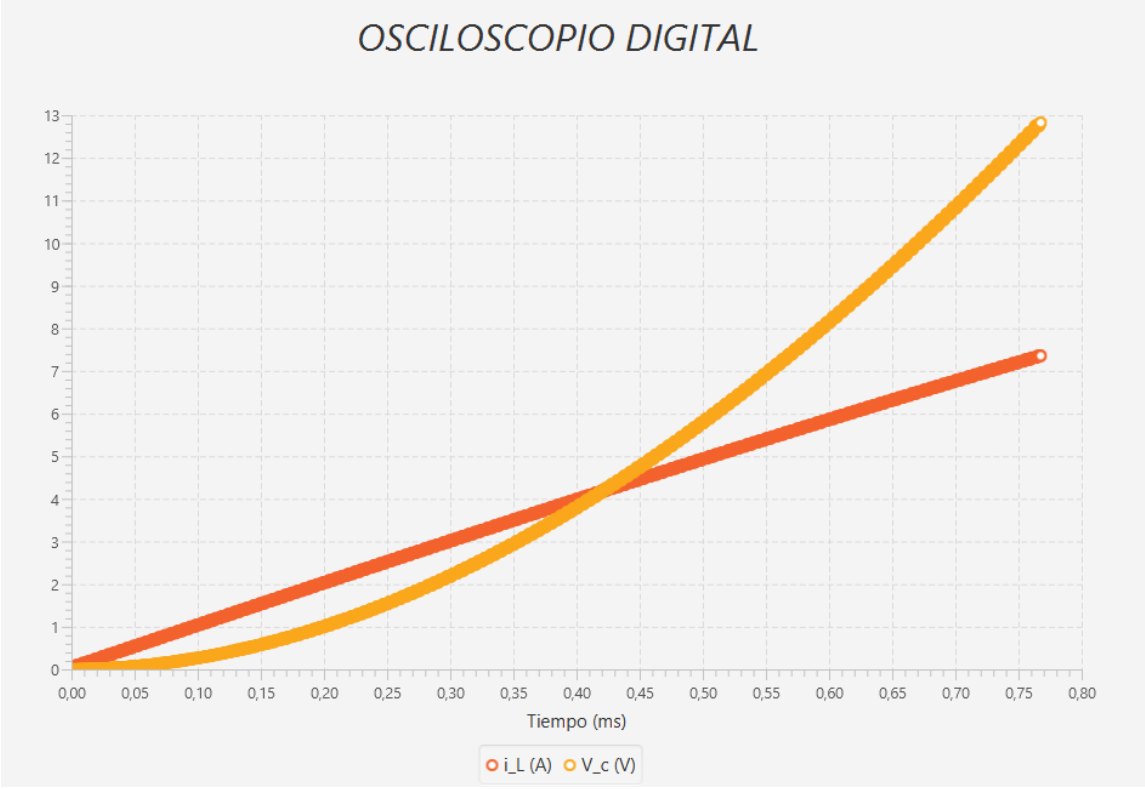

Figura 6-25. Simulación hardware con frecuencia del interruptor = 200 kHz

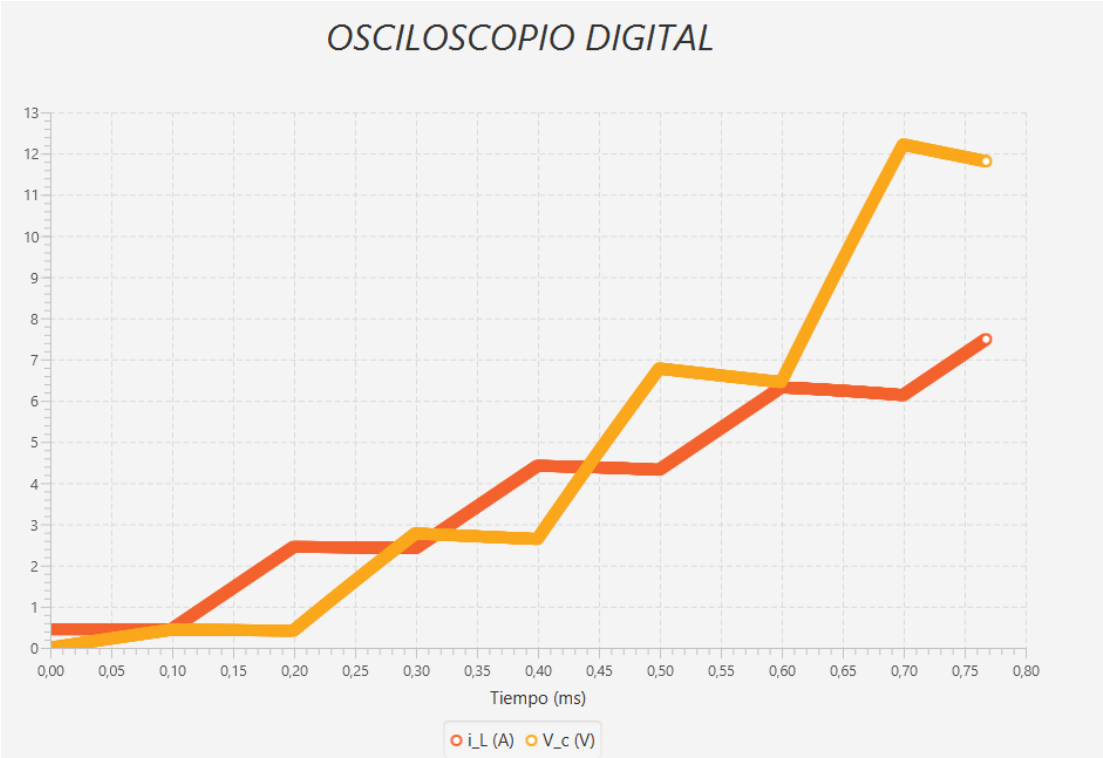

Figura 6-26. Simulación hardware con frecuencia del interruptor = 5 kHz

En las Figuras 6-24, 6-25 y 6-26 se puede comprobar lo indicado anteriormente. Cuanto más alta sea la frecuencia del interruptor, menor será el rizado que posea el circuito. De hecho se puede observar como en la Figura 6-25 que se corresponde con la frecuencia más alta, apenas se observa rizado en las señales. También se observa que cuanto más baja sea la frecuencia del interruptor mayores son los saltos que se producen en las variables de estado debido al rizado.

7 Conclusiones y trabajo futuro

7.1 Conclusiones

Las técnicas HIL han supuesto un gran avance en la simulación en tiempo real de múltiples sistemas, entre los que se encuentran los convertidores de potencia. Estos sistemas integran elementos hardware para acelerar la simulación de las plantas. La irrupción de las FPGAs en estos sistemas ha supuesto una gran revolución en la última década.

En este trabajo se ha diseñado e implementado un sistema HIL para simular convertidores de potencia, centrándose en el convertidor Flyback, aunque los principios de diseño son genéricos. Para ello, se ha diseñado un modelo digital del convertidor de potencia y se ha implementado en una FPGA (Arty Z7-20). Gracias a la FPGA, se pueden realizar cálculos en paralelo para así conseguir un simulador en tiempo real con pasos de integración de decenas de nanosegundos.

En este trabajo se ha desarrollado un sistema HIL completo. Es decir, no sólo se ha diseñado e implementado el modelo digital del convertidor, sino que también se le ha dotado al sistema de comunicación estable con una aplicación de ordenador. Además, se ha implementado un capturador de datos digital (muy similar a un osciloscopio) que permite ser configurado para que el usuario extraiga los resultados de la simulación, permitiendo el uso de *triggers*.

En las pruebas del sistema, se ha comprobado el correcto funcionamiento del modelo digital del convertidor. Para ello, se han hecho pruebas comparando un modelo con variables de tipo *real* y otro con aritmética de tipo float, obteniendo resultados idénticos. Para comprobar el funcionamiento del modelo se ha modificado el valor del ciclo de trabajo de la señal del interruptor para ver la influencia de éste en el circuito, y compararlo con las ecuaciones teóricas del convertidor.

También se han realizado pruebas de integración, probando tanto el simulador como la comunicación con la aplicación de ordenador. Con dicha aplicación se ha conseguido configurar el modelo del convertidor y ver el resultado de las señales en el osciloscopio digital insertado en la aplicación. Entre otras pruebas, se han probado varias combinaciones de *trigger* posibles para observar el comportamiento de las señales a partir de un punto determinado y el correcto funcionamiento del *trigger*.

Como se ha mostrado en la sección 6.1, la cantidad de recursos utilizada por el periférico se situaba por debajo del 15% en todos los recursos. Para una placa que cuesta aproximadamente 190 €, es un porcentaje muy bajo. Por tanto, se podría usar este proyecto para añadir más convertidores y dar la posibilidad de interconectarlos con el convertidor Flyback ya existente.

7.2 Trabajo futuro

Como líneas de futuro se propone implementar otros modelos de convertidor e intentar interconectarlos. Otra opción sería la de poder elegir el convertidor que se desea usar en la aplicación de ordenador y con ello conseguir tener una biblioteca con muchos convertidores para así poder observar el comportamiento de cada uno. Otra mejora sería integrar DACs para poder extraer los datos generados por el modelo.

Si en un futuro se decidiera comercializar, se recomendaría mejorar la visualización de los datos en el osciloscopio digital, mediante la posibilidad de hacer zoom sobre la gráfica y tener la posibilidad de tener un cursos que facilite el valor directamente desde la pantalla de visualización. Finalmente, sería de gran ayuda tener la opción de extraer el valor de las señales mediante CSV u otro formato, con el fin de tratarlos en Excel o MATLAB.

Referencias

[1] B. J. Patella, A. Prodic, A. Zirger, and D. Maksimovic, “High-frequency digital PWM controller IC for DC-DC converters,” IEEE Trans. Power Electron., vol. 18, no. 1, pp. 438–446, Jan. 2003.

[2] Z. Tir, O. P. Malik, and A. M. Eltamaly, “Fuzzy logic based speed control of indirect field oriented controlled double star induction motors connected in parallel to a single six-phase inverter supply,” Electr. Power Syst. Res., vol. 134, pp. 126–133, May 2016

[3] Ned Mohan, Tore M. Undeland, William P.Robbins. Electrónica de Potencia. Convertidores, aplicaciones y diseño. 3º Edición – 2009.º

[4] Sandra Jurado Jabonero. Emulación HIL de convertidores conmutados utilizando modelos parametrizables. Trabajo fin de máster, Universidad Autónoma de Madrid, Junio 2016.

[5] Rafael Ramos Lara, Sistemas digitales de control en tiempo discreto, Departamento de Ingeniería electrónica. Universidad Politécnica de Cataluña. Febrero 2007

[6] A. de Castro, Aplicación del Control Digital Basado en Hardware Específico para Convertidores de Potencia Conmutados, PhD thesis, Universidad Politécnica de Madrid, 2003.

[7] P. Zumel, M. García-Valderas, A. Lázaro, C. Loópez-Ongil, and A. Barrado, Cosimulation psim-modelsim oriented to digitally controlled switching power Converters, in Control and Modeling for Power Electronics (COMPEL), 2010 IEEE 12th Workshop, Junio 2010.

[8] A. de Castro, T. Riesgo, O. Garcia, and R. Prieto, Comparing vhdl and vhdl-ams for modelling and simulation of power converters with digital control, in XVIII Conference on Design of Circuits and Integrated Systems (DCIS), Nov. 2003.

[9] A. Sanchez, A. de Castro and J. Garrido, "Parametrizable Fixed-Point Arithmetic for HIL With Small Simulation Steps," in IEEE Journal of Emerging and Selected Topics in Power Electronics, vol. 7, no. 4, pp. 2467-2475, Dec. 2019, doi: 10.1109/JESTPE.2018.2886908.

[10] A. S. Vijay, S. Doolla, and M. C. Chandorkar, “Real-time testing approaches for microgrids,” IEEE J. Emerg. Sel. Topics Power Electron., vol. 5, no. 3, pp. 1356–1376, Sep. 2017

[11] B. Lu, X. Wu, H. Figueroa, and A. Monti, “A low-cost real-time hardware-in-the-loop testing approach of power electronics controls,” IEEE Trans. Ind. Electron., vol. 54, no. 2, pp. 919–931, Apr. 2007

[12] M. Matar and R. Iravani, “FPGA implementation of the power electronic converter model for real-time simulation of electromagnetic transients,” IEEE Trans. Power Del., vol. 25, no. 2, pp. 852–860, Apr. 2010.

[13] X. Yang, C. Yang, T. Peng, Z. Chen, B. Liu, and W. Gui, “Hardwarein-the-loop fault injection for traction control system,” IEEE J. Emerg. Sel. Topics Power Electron., vol. 6, no. 2, pp. 696–706, Jun. 2018.

[14] L. Ibarra, A. Rosales, P. Ponce, A. Molina, and R. Ayyanar, “Overview of real-time simulation as a supporting effort to smart-grid attainment,” Energies, vol. 10, no. 6, 2017, Art. no. 817. [Online]. Available: http://www.mdpi.com/1996-1073/10/6/817

[15] S. Srdic, X. Liang, C. Zhang, W. Yu, and S. Lukic, “A SiC-based highperformance medium-voltage fast charger for plug-in electric vehicles,” in Proc. IEEE Energy Convers. Congr. Expo. (ECCE), Sep. 2016, pp. 1–6

[16] A. Sanchez, A. de Castro and J. Garrido, "A Comparison of Simulation and Hardware-in-the- Loop Alternatives for Digital Control of Power Converters," in IEEE Transactions on Industrial Informatics, vol. 8, no. 3, pp. 491-500, Aug. 2012, doi: 10.1109/TII.2012.2192281.

[17] O. Lucia, I. Urriza, L. Barragan, D. Navarro, O. Jimenez, and J. Burdio, “Real-time FPGA-based hardware-in-the-loop simulation test bench applied to multiple-output power converters,” IEEE Trans. Ind. Appl., vol. 47, no. 2, pp. 853–860, Mar.–Apr. 2011.

[18] IEEE Standard for Floating-Point Arithmetic," in IEEE Std 754-2019 (Revision of IEEE 754-2008), vol., no., pp.1-84, 22 July 2019

[19] https://static.docs.arm.com/ihi0022/h/IHI0022H_amba_axi_protocol_spec.pdf

[20] https://www.realdigital.org/doc/a9fee931f7a172423e1ba73f66ca4081

[21] https://static.docs.arm.com/ihi0022/h/IHI0022H_amba_axi_protocol_spec.pdf

[22] http://fpgasite.blogspot.com/2017/07/xilinx-axi-stream-tutorial-part-1.html

[23] https://www.xilinx.com/support/documentation/ip_documentation/axi_dma/v7_1 /pg021 _axi_dma.pdf

[24] https://www.rfc-es.org/rfc/rfc1180-es.txt

Glosario

FPGA	*Field-Programmable Gate Array*
VHDL	*VHSIC Hardware Description Languaje*
DSP	*Digital Signal Processor*
ADC	*Analog-to-Digital Conversion*
DAC	*Digital-to-Analog Conversion*
FF	*Flip Flop*
HIL	*Hardware In The Loop*
SoC	*System on Chip*
DMA	*Direct Memory Access*
CC	*Corriente Continua*
CA	*Corriente Alterna*